Ս. Ս. ԱՎԵՏԻՍՅԱՆ, Ս. Վ. ԴԱՆԻԵԼՅԱՆ

ԻՆՖՈՐՄԱՏԻԿԱ

12-րդ դասարան

ՀԱՆՐԱԿՐԹԱԿԱՆ ԱՎԱԳ ԴՊՐՈՑԻ
ԸՆԴՀԱՆՈՒՐ ԵՎ ՀՈՒՄԱՆԻՏԱՐ ՀՈՍՔԵՐԻ ՀԱՄԱՐ

ՀԱՍՏԱՏՎԱԾ Է ՀՀ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ
ՆԱԽԱՐԱՐՈՒԹՅԱՆ ԿՈՂՄԻՑ

ԵՐԵՎԱՆ
ՏԻԳՐԱՆ ՄԵԾ
ՀՐԱՏԱՐԱԿՉՈՒԹՅՈՒՆ ՓԲԸ
2012

ՀՏԴ 373.167.1 : 004 (075.3)
ԳՄԴ 73 ց72
Ա 791

Մասնագիտական խմբագիր՝ Ռ. Վ. Աղգաշյան

Ավետիսյան Ս.Ս.
Ա 791 Ինֆորմատիկա: 12-րդ դաս. դասագիրք. հանրակրթական ավագ դպրոցի ընդհանուր և հումանիտար հոսքերի համար / Ս. Ս. Ավետիսյան, Ս. Վ. Դանիելյան; Մասն. խմբ.՝ Ռ. Աղգաշյան. – Եր.: Տիգրան Մեծ, 2012. – 128 էջ:

ՀՏԴ 373.167.1 : 004 (075.3)
ԳՄԴ 73 ց72

ISBN 978-99941-0-492-5

ՆԵՐԱԾՈՒԹՅՈՒՆ

Ինֆորմատիկայի այս դասագիրքը շարունակելու է հարստացնել համակարգչին առնչվելու ձեր հմտությունները. շարունակելու եք աղյուսակային տվյալների հետ սկսած ձեր աշխատանքը, ուսումնասիրելու եք հենքային տվյալների ղեկավարման *Microsoft Access* համակարգի հնարավորությունները:

Շարունակելու եք նաև զարգացնել ալգորիթմներ կազմելու ձեր արդեն ձեռք բերած հմտությունները. խնդիրների լուծման ալգորիթմները համակարգչով իրականացնելու համար կծանոթանաք ուսուցողական նպատակով ստեղծված և ուսուցման ոլորտում մեծ կիրառություն գտած ծրագրավորման *Պասկալ* լեզվի որոշ հնարավորություններին:

Հուսով ենք՝ Համացանցային կայքեր ստեղծելու գործընթացը հետաքրքիր է ձեզ համար, քանի որ շարունակելու եք ծանոթանալ դրա նոր, առավել գործուն հնարավորություններին:

Դասագրքում նոր տերմինները տպագրված են հիմնական տեքստից տարբեր գույնով, իսկ առավել կարևոր և առանցքային հասկացությունները շրջանակների մեջ են առնված:

Դասի հիմնական նյութին հաջորդող *Օգտակար է իմանալ* խորագիրը կրող տեղեկատվությունը շարունակելու է օգնել ձեզ՝ համակողմանի յուրացնելու համապատասխան նյութը: Դասագրքի *Հարցեր և առաջադրանքներ* բաժինը կօգնի ձեզ պարզելու, թե որքանով եք հասկացել դասը: Որոշակի գործնական հմտություններ ձեռք բերելու համար ձեզ կօգնեն *Լաբորատոր աշխատանքները*, որոնք մանրամասն ցուցումներ են պարունակում:

1. ՏՎՅԱԼՆԵՐԻ ՀԵՆՔԵՐ

§1.1. ՏՎՅԱԼՆԵՐԻ ՀԵՆՔԻ ՍՏԵՂԾՈՒՄ

8-րդ և *9*-րդ դասարանների դասընթացից ծանոթ եք տվյալների հենքերի ղեկավարման *Microsoft Access* համակարգի *2003* տարբերակի որոշ հնարավորությունների: Այժմ փորձենք ուսումնասիրել առավել արդի *Microsoft Access 2007* համակարգը:

Համակարգի միջավայր մտնելու համար պետք է մկնիկի օգնությամբ հաջորդաբար իրականացնել հետևյալ քայլերը.

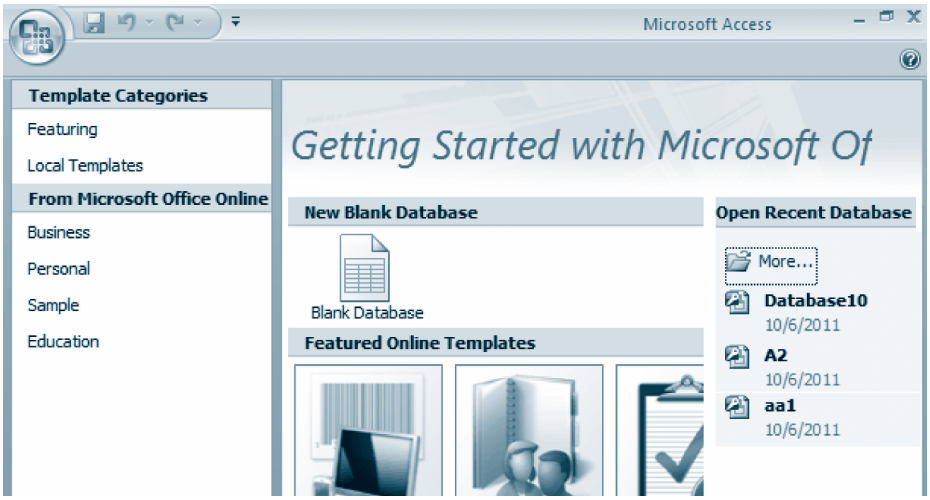
Start ⇨ All Programs ⇨ Microsoft Office ⇨ Microsoft Office Access 2007

Այժմ *Access*-ը սպասում է հետևյալ հրահանգներից որևէ մեկին.

- գոյություն ունեցող որևէ հենք բացել,
- նոր հենք ստեղծել:

Տվյալների որևէ հենք բացելու համար անհրաժեշտ է.

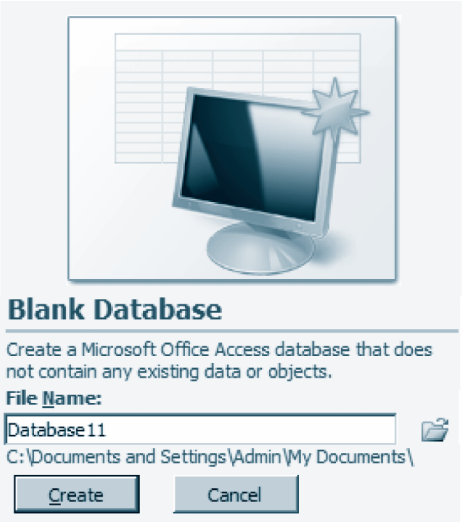
- բացված պատուհանի (նկ. 1.1.) աջ մասում տեղակայված *Open Recent Database* բաժնում ներառված տվյալների հենքերի ցուցակից ընտրել անհրաժեշտը,
- եթե փնտրված ֆայլը բացված ցուցակում չի ընդգրկված, ապա որոնումը շարունակել More... կոճակի օգնությամբ:



Նկ. 1.1. Microsoft Access 2007-ի աշխատանքի նախնական պատուհան

Տվյալների նոր հենք ստեղծելու համար անհրաժեշտ է.

- ընտրել *New Blank Database* բաժնի *Blank Database* կոճակը (նկ. 1.1.)
- բացված տիրույթի (նկ. 1.2.) *File Name* դաշտում նշել ստեղծվող հենքի անվանումը (նկարում բերված օրինակում՝ *Database 11)*: Ստեղծվող ֆայլի պահպանման հասցեն ընտրել կոճակով բացված պատուհանում,
- սեղմել *Create* կոճակը:



Նկ. 1.2. Նոր ֆայլ ստեղծելու փիրույթ

Տվյալների հենք կարելի է ստեղծել նաև *Template Categories* խմբի *From Microsoft Office Online* հրամանով (նկ. 1.1.) բացվող շաբլոններից որևէ մեկի օգնությամբ:

Ms Access 2007 համակարգի հիմնական աշխատանքային պատուհանն ունի նկ. 1.3.-ում բերված տեսքը:

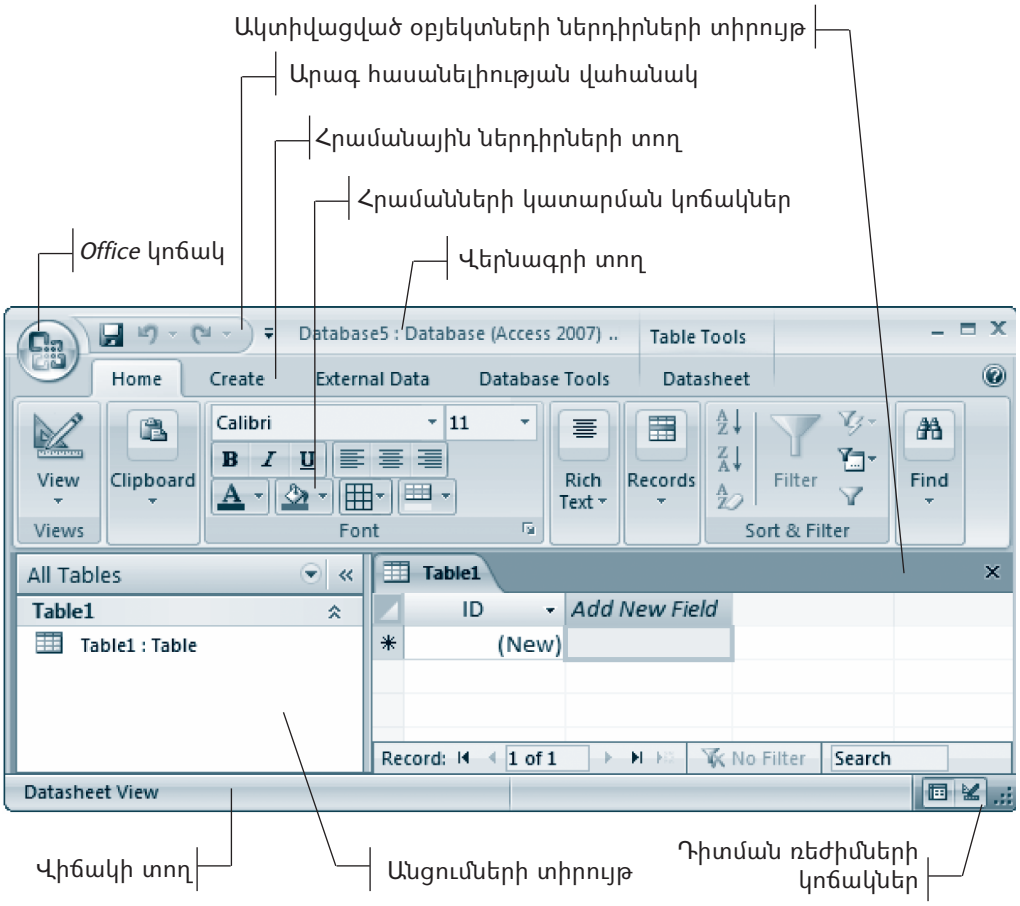
Ծանոթանանք պատուհանի հիմնական բաղադրիչներին:

Արագ հասանելիության վահանակը նախատեսված է առավել հաճախ կիրառվող հրամանների կոճակների համար:

Հրամանային ներդիրների փողը 5 ներդիր է պարունակում: Նկ. 1.3-ում բացված է *Home* ներդիրը: Ինչպես երևում է նկարից՝ այն յոթ խումբ է ընդգրկում. *View, Clipboard, Font, Rich Text, Records, Sort & Filter, Find*, որոնցից յուրաքանչյուրն իր հերթին այլ հրամանների կոճակներ է ներառում:

Վերնագրի փողում արտածվում է տվյալ պահին խմբագրիչի աշխատանքային տիրույթում առկա փաստաթղթի անվանումը (նկ. 1.3.-ում՝ *Database5*):

Վիճակի փողում ինֆորմացիա է արտածվում տվյալների թողարկված հենքի և կատարվող գործողությունների մասին:

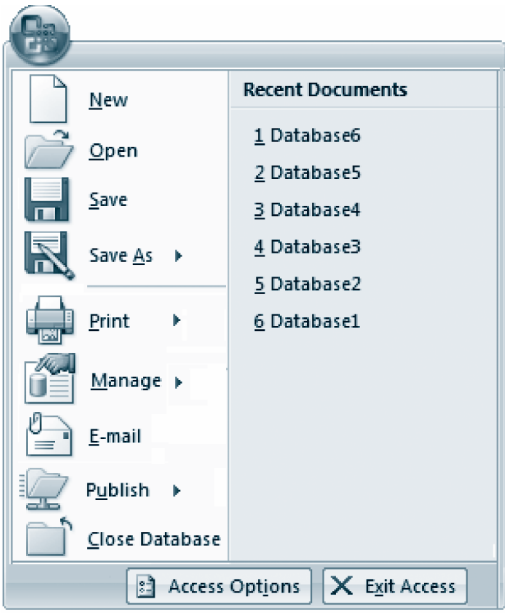


Նկ. 1.3. Microsoft Access 2007 համակարգի պատուհան

Անցումների տիրույթում արտածվում են տվյալների հենքի օբյեկտները, ինչը հնարավորություն է տալիս ընտրել դրանցից անհրաժեշտը:

Ակտիվացված օբյեկտների ներդիրների տիրույթը ներառում է տվյալների հենքի ակտիվացված օբյեկտների անվանումները, ինչը ևս հնարավորություն է տալիս ընտրելու դրանցից անհրաժեշտը:

Office կոճակով բացվում է ֆայլերի հետ առավել հաճախ կիրառվող հրամանների մենյուն (նկ. 1.4.), որի բաղադրիչներին կծանոթանաք հետագայում:



Նկ. 1.4. Office կոճակի պատուհան

Ms Access-ի աշխատանքը կարելի է ավարտել համակարգի հիմնական պատուհանի փակման կոճակով:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *Access 2007*-ով ստեղծված ֆայլերն ունեն *.mdbx* ընդլայնումը:
- Տվյալների հենքում հայատառ ինֆորմացիա ներմուծելու նպատակով անհրաժեշտ է ընտրել հայկական տառատեսակի *Unicode* տարբերակը:

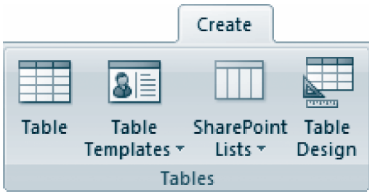
1. ***Microsoft Access* համակարգի հիմնական աշխատանքային պատուհանի ինչ հիմնական տիրույթներ գիտեք:**
2. **Ինչպե՞ս կարելի է համակարգչում առկա որևէ հենք բացել:**
3. **Ինչպե՞ս են տվյալների նոր հենք ստեղծում:**

§1.2. ԱՂՅՈՒՍԱԿՆԵՐԻ ՍՏԵՂԾՈՒՄ ԵՎ ԽՄԲԱԳՐՈՒՄ

Աղյուսակը տվյալների հենքի առաջին և հիմնական օբյեկտն է: Տվյալների հենքը կարող է մեկ կամ մի քանի աղյուսակ ունենալ:

Աղյուսակ ստեղծելու համար պետք է ընտրել *Create* (ստեղծել) ներդիրի *Tables* խմբի հետևյալ չորս տարրերից անհրաժեշտը (նկ. 1.5.).

- *Table* – աղյուսակի ստեղծում՝ *տվյալների ներմուծմամբ,*
- *Table Templates* – աղյուսակի ստեղծում՝ *շաբլոնների օգնությամբ,*
- *SharePoint Lists* – աղյուսակի ստեղծում՝ *SharePoint*-ի օգնությամբ,
- *Table Design* – աղյուսակի ստեղծում՝ *կոնստրուկտորի* օգնությամբ:



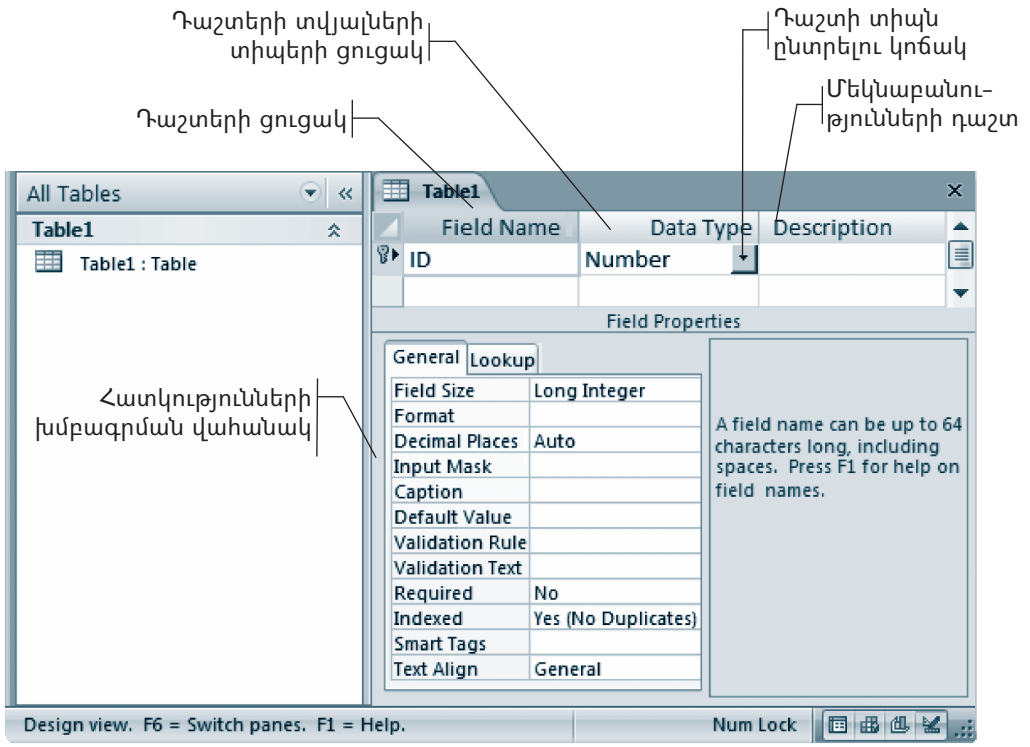
Նկ. 1.5. Աղյուսակ ստեղծելու միջոցներ

Տվյալների ներմուծմամբ աղյուսակ ստեղծելու տարբերակն ընտրելիս համակարգը թույլատրում է տվյալներն աղյուսակ ներմուծել այնպես, ինչպես ձեզ ծանոթ *Excel* էլեկտրոնային աղյուսակում:

Շաբլոնների օգնությամբ աղյուսակ ստեղծելու տարբերակն ընտրելու դեպքում հնարավորություն եք ստանում տիպային աղյուսակ ձևավորել. այստեղ կարևորն աղյուսակի նմուշը ճիշտ ընտրելն է:

SharePoint-ի օգնությամբ աղյուսակ կարելի է ստեղծել, երբ *Microsoft Office SharePoint Services* կայքից օգտվելու հնարավորություն կա: Այս դեպքում կարելի է *SharePoint* ցուցակը *Access*-ի աղյուսակ ներմուծել:

Աղյուսակներ ստեղծելու գործընթացին առավել հանգամանորեն ծանոթանալու նպատակով ուսումնասիրենք այդ նպատակի համար նախատեսված տարրերից վերջինը՝ *կոնստրուկտորի օգնությամբ աղյուսակ ստեղծելը*: Կոնստրուկտորի պատուհանը (նկ. 1.6.) աղյուսակի կառուցվածք ստեղծելու և խմբագրելու բլանկ է ներկայացնում: Այն բաղկացած է երեք սյուներից՝ *Field Name* (դաշտի անվանում), *Data Type* (տվյալների տիպ) և *Description* (նկարագրություն): Նախ անհրաժեշտ է *Field Name* սյան մեջ լրացնել անհրաժեշտ դաշտերի անվանումները, իսկ *Data Type* սյան մեջ ընտրել դրանց տիպերը, այսինքն՝ այդ դաշտերում պահպանվելիք տվյալների ձևաչափերը: Անհրաժեշտության դեպքում *Description* սյան մեջ կարելի է յուրաքանչյուր դաշտին առնչվող մեկնաբանություններ տալ:



Նկ. 1.6. Աղյուսակի կառուցվածքը նախագծելու պատուհան

Դաշտի տվյալների տիպն ընտրելու համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել դաշտի աջ մասում տեղակայված սլաքի վրա ու սեղմել ձախ սեղմակը,
- առաջարկվող ցուցակից (նկ. 1.7.) ընտրել անհրաժեշտ տիպը:

Դաշտի տվյալների որոշ տիպերի մենք ծանոթ ենք էլեկտրոնային աղյուսակներից: Տվյալների հենքերը, որպես կանոն, հնարավորություն են տալիս աշխատել տվյալների առավել մեծաքանակ տիպերի հետ: Ծանոթանանք *Access*-ում կիրառվող հետևյալ տիպերին.

Text (*տեքստային*) – տվյալների այս տիպն ունեցող դաշտը կարող է մինչև *255* պայմանանշան պարունակող տեքստ ներառել:

Memo (*տեքստային*) – տվյալների այս տիպն ունեցող դաշտը կարող է մինչև *65535* պայմանանշան պարունակող տեքստ ներառել:

Number (*թվային*) – կարող է ցանկացած թիվ պարունակել:

Date/Time (*ամսաթիվ/ժամանակ*) – կարող է տվյալներ պարունակել ամսաթվի և ընթացիկ ժամանակի վերաբերյալ:

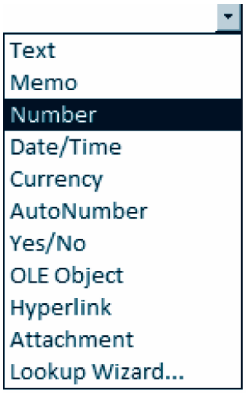
Currency (*դրամական*) – կարող է թիվ պարունակել, որի ամբողջ մասը հնարավորություն ունի մինչև *15*, իսկ կոտորակային մասը՝ մինչև *4* նիշ պահել:

AutoNumber (*հաշվիչ*) – կարող է բնական թիվ պարունակել, որի արժեքը յուրաքանչյուր հաջորդ գրառմանն անցնելիս ավտոմատ մեկով ավելացվում է:

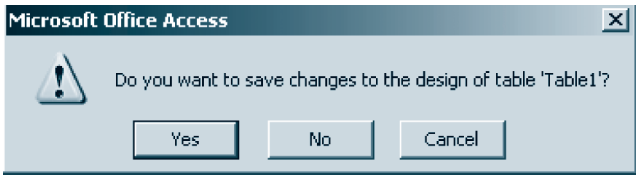
Yes/No (*տրամաբանական*) – կարող է պարունակել *Այո* կամ *Ոչ* արժեքներից որևէ մեկը:

Յուրաքանչյուր դաշտ բնութագրվում է հատկանշական պարամետրերով, որոնք սահմանում են դաշտում պահպանվող տվյալների մշակման ու պահպանման մեթոդները: Այս պարամետրերը տեղակայվում են հատկությունների խմբագրման վահանակում (նկ. 1.6.): Նման պարամետրերի միջոցով կարելի է սահմանել տեքստային դաշտի երկարությունը, թվային տվյալների տիպը (ամբողջ կամ իրական) և այլն:

Աղյուսակի կառուցվածքը սահմանելուց հետո պատուհանի ղեկավարման  սեղմակով պետք է փակել *Աղյուսակի կառուցվածքի նախագծում* պատուհանը: Սրան ի պատասխան՝ համակարգը աղյուսակի պահպանման վերաբերյալ երկխոսական պատուհան է բացում (նկ. 1.8.):



Նկ. 1.7. Դաշտի տվյալների տիպերի ցուցակ

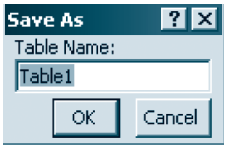


Նկ. 1.8. Երկխոսական պատուհան՝ աղյուսակը պահպանելու վերաբերյալ

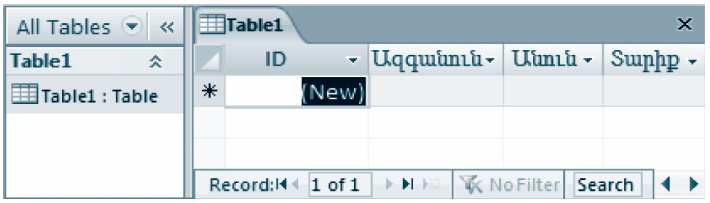
Բերված պատուհանից կարելի է հրաժարվել *Cancel*-ի ընտրմամբ: Եթե ընտրվի *No* կոճակը, ապա ստեղծված աղյուսակը չի պահպանվի, իսկ *Yes*-ն ընտրելիս մի նոր պատուհան կբացվի (նկ. 1.9.), որի *Table Name* դաշտ պետք է ներմուծել ստեղծվող աղյուսակի անվանումն ու սեղմել *OK* կոճակը: Եթե համակարգի կողմից առաջարկվող անվանումը (օրինակ՝ *Table1*) բավարար է, ապա պետք է սեղմել *OK* կոճակը՝ առանց նոր անվանում ներմուծելու:

Սրեղծված աղյուսակը բացելու համար անհրաժեշտ է մկնիկի ցուցիչը տեղադրել անցումների տիրույթում արտածված աղյուսակի անվան վրա (նկ. 1.3.) ու մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարել:

Նոր ստեղծված աղյուսակը գրառումներ չունի. այն պարունակում է միայն աղյուսակը բնութագրող սյուների անվանումները (նկ. 1.10.):



Նկ. 1.9. Աղյուսակի անվանումը ներմուծելու պատուհան



Նկ. 1.10. Նոր աղյուսակի օրինակ

Աղյուսակում *տվյալների ներմուծումն* իրականացվում է սովորական կարգով: Ներմուծման համար ցուցիչը մկնիկի կամ ←, ↑, →, ↓ ստեղների օգնությամբ տեղադրվում է անհրաժեշտ բջիջում: Ընդարձակ աղյուսակների հետ աշխատելիս կարելի է օգտվել պատուհանի ստորին մասում տեղադրված անցման կոճակների վահանակի գործիքներից:

Տվյալների ներմուծումն ավարտելիս արդյունքներն ավտոմատ պահպանվում են: Եթե աշխատելու ընթացքում աղյուսակի պարամետրերից թեկուզ մեկը (օրինակ՝ դաշտի լայնությունը) փոփոխման է ենթարկվել, ապա համակարգն այդ փոփոխությունները պահպանելու վերաբերյալ լրացուցիչ հարցադրում կանի:

Աղյուսակի կառուցվածքը խմբագրելու համար անհրաժեշտ է.

- *Home* ներդիրի *Views* խմբի 

գործիքի ստորին մասի ▼ սլաքով բացված վահանակից ընտրել  (*Design View*) գործիքը,
- բացված պատուհանում կատարել անհրաժեշտ խմբագրումը:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Նոր ստեղծվող աղյուսակի մեջ *Access*-ը *ID* անունով դաշտ է ստեղծում, որը *AutoNumber* տիպի է (աղյուսակում նոր գրառում ավելացնելիս այս դաշտի պարունակությունը մեկով ավելացվում է):
- Դաշտի չափերը փոխելու համար անհրաժեշտ է մկնիկի ցուցիչը տեղադրել դաշտի աջ կամ ձախ սահմանի վրա և այն խաչի տեսք ստանալիս՝ մկնիկի ձախ սեղմակով անհրաժեշտ փոփոխություններն իրականացնել:
- Դաշտի չափերն ավտոմատ կարգավորելու համար անհրաժեշտ է մկնիկի ցուցիչը տեղադրել դաշտի աջ կամ ձախ սահմանի վրա և այն խաչի տեսք ստանալիս՝ մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարել:

1. *Access*-ում կիրառվող դաշտերի ինչպիսի՞ տիպեր գիտեք:
2. Ինչո՞վ են իրարից տարբերվում *Text* և *Memo* տիպերը:
3. Կարելի՞ է արդյոք խմբագրել արդեն ստեղծված աղյուսակը:

Լաբորատոր աշխատանք 1.1
Աղյուսակի սպեղծում

1. Հաջորդաբար կատարելով *Start, All Programs, Microsoft Office, Microsoft Office Access 2007* քայլերը՝ *Access 2007*-ի միջավայր մտեք:
2. Բացված պատուհանում ընտրեք *New Blank Database* բաժնի *Blank Database* կոճակը:
3. Ակտիվացած տիրույթի *File Name* դաշտում ներմուծեք ստեղծվող տվյալների հենքի անվանումը՝ *Lab_1_1_**, որտեղ *–ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:
4. կոճակով բացված պատուհանում ընտրեք տվյալ դասարանին հատկացված թղթապանակը:
5. Սեղմեք *Create* կոճակը:
6. *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքով բերված ցուցակից ընտրեք *Design View* հրամանը:
7. Ընդունեք աղյուսակը պահպանելու համար առաջարկվող *Table1* անվանումն ու սեղմեք *OK* կոճակը:
8. Ընտրեք *Arial Armenian Unicode* տառատեսակը:
9. Կոնստրուկտորի պատուհանի առաջին տողի *Field Name* դաշտում ներմուծեք *Ազգանուն* բառը և նկատեք, որ այդ տողի *Data Type* դաշտում ավտոմատ ընտրվեց *Text* տիպը:
10. Երկրորդ տողի *Field Name* դաշտում ներմուծեք *Անուն* բառն ու *Data Type* դաշտում ընտրեք *Text* տիպը:
11. Երրորդ տողի *Field Name* դաշտում ներմուծեք *Մաթեմատիկա* բառը:

12. Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
13. Չորրորդ տողի *Field Name* դաշտում ներմուծեք *Ֆիզիկա* բառը:
14. Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
15. Հինգերորդ տողի *Field Name* դաշտում ներմուծեք *Ինֆորմատիկա* բառը:
16. Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
17. Աղյուսակի կառուցվածքը ստեղծելուց հետո պատուհանի ղեկավարման սեղմակով փակեք պատուհանը:
18. Աղյուսակի պահպանման մասին տրված հարցմանը պատասխանեք *Yes*:
19. Համաձայնեք առաջարկվող *Table1* անվանմանն ու *OK* կոճակը սեղմելով համոզվեք, որ պատուհանի ձախ վահանակին *Table1* անվանումով աղյուսակ առաջացավ:
20. Մկնիկի ձախ սեղմակի կրկնակի սեղմումով բացեք *Table1* աղյուսակը:
21. Ստորև բերված աղյուսակի օրինակով *Ազգանուն* և *Անուն* դաշտերում դասարանի *5* աշակերտների անուն, ազգանուն ներմուծեք, իսկ *Ինֆորմատիկա, Մաթեմատիկա* և *Ֆիզիկա* դաշտերում ներմուծեք տվյալ առարկաներից նրանց կիսամյակային գնահատականները.

ID	Field1	Field2	Field3	Field4	Field5
1	Ազգանուն	Անուն	Մաթեմատիկա	Ֆիզիկա	Ինֆորմատիկա
2	Ամիրյան	Արա	9	10	8
3	Բաբայան	Անահիտ	8	8	9
4	Էմինյան	Գառնիկ	7	8	7

22. Սեղմեք պատուհանի ղեկավարման սեղմակն ու աղյուսակի պահպանմանն ուղղված հարցմանը պատասխանեք *Yes*:
23. Ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման սեղմակից:

§1.3. ԱՇԽԱՏԱՆՔ ԱՂՅՈՒՍԱԿԻ ԴԱՇՏԵՐԻ ԵՎ ԳՐԱՌՈՒՄՆԵՐԻ ՀԵՏ

Այժմ առավել հանգամանորեն ծանոթանանք աղյուսակի բաղադրիչների հետ կապված աշխատանքին: Նախ սովորենք *նշել* աղյուսակի բաղադրիչները կազմող դաշտերն ու գրառումները:

Աղյուսակի *որևէ դաշտ նշելու* համար (նկ. 1.11.) անհրաժեշտ է մկնիկի ցուցիչը տեղադրել համապատասխան դաշտի անվան վրա և, երբ ցուցիչը կձևափոխվի սլաքի, սեղմել ձախ սեղմակը:

Աղյուսակի *որևէ գրառում նշելու* համար (նկ. 1.12.) անհրաժեշտ է մկնիկի ցուցիչը տեղադրել գրառման ձախ եզրային վանդակի վրա և, երբ ցուցիչը կձևափոխվի սլաքի, սեղմել ձախ սեղմակը:

ID	aaa	bbb	ddd
1	1	2	3
2	5	2	6
3	5	4	3
4	3	1	4

Նկ. 1.11. Նշված դաշտով աղյուսակ

ID	aaa	bbb	ddd
1	1	2	3
2	5	2	6
3	5	4	3
4	3	1	4

Նկ. 1.12. Նշված գրառումով աղյուսակ

Աղյուսակի *նշված գրառումը* կամ *դաշտը կարելի է պատճենել, տեղափոխել, հեռացնել* *Home* ներդիրի *Clipboard* խմբի գործիքների վահանակի 

, 

և 

գործիքների օգնությամբ, ինչպես նաև նշված դաշտի կամ գրառման վրա մկնիկի աջ սեղմակով բացված ենթատեքստային մենյուի համապատասխան հրամաններով:

Աղյուսակի *դաշտն անվանափոխելու* համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել դաշտի անվան վրա և սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից (նկ. 1.13.) ընտրել *Rename Column* հրամանը,
- նախկին անվան փոխարեն ներմուծել նոր անվանումը:

Աղյուսակում *նոր դաշտ ավելացնելու* համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել այն դաշտի անվան վրա, որին պետք է նախորդի ավելացվող դաշտն ու սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից (նկ. 1.13.) ընտրել *Insert Column* հրամանը:

Աղյուսակից *դաշտ հեռացնելու* համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել հեռացման ենթակա դաշտի անվան վրա ու սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից (նկ. 1.13.) ընտրել *Delete Column* հրամանը:

Աղյուսակի *դաշտը թաքցնելու* համար անհրաժեշտ է.

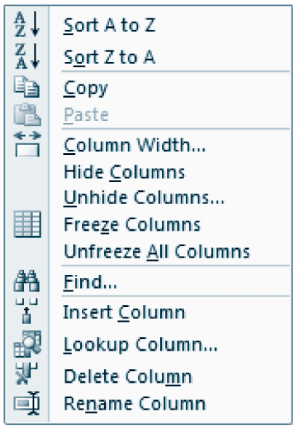
- մկնիկի ցուցիչը տեղադրել տվյալ դաշտի անվան վրա ու սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից (նկ. 1.13.) ընտրել *Hide Columns* հրամանը:

Աղյուսակի *թաքցված դաշտը տեսանելի դարձնելու* համար անհրաժեշտ է.

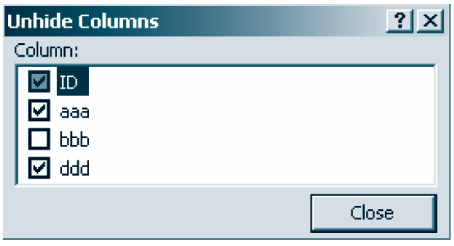
- մկնիկի ցուցիչը տեղադրել դաշտերից որևէ մեկի անվան վրա ու սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից (նկ. 1.13.) ընտրել *Unhide Columns* հրամանը,
- բերված պատուհանում (նկ. 1.14.) ընտրել անհրաժեշտ դաշտն ու սեղմել *Close* կոճակը:

Աղյուսակի *թվային տիպի դաշտի պարունակությունն ըստ աճման կամ նվազման կարգավորելու* համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել տվյալ դաշտի որևէ բջիջի վրա ու սեղմել աջ սեղմակը,
- բացված ենթատեքստային մենյուից ընտրել  (*Sort A to Z*) հրամանն ըստ աճման և  (*Sort Z to A*) հրամանն ըստ նվազման կարգավորելու նպատակով:



Նկ. 1.13. Դաշտին առնչվող ենթատեքստային մենյու



Նկ. 1.14. Թաքցված դաշտերից անհրաժեշտը ցուցադրելու վահանակ

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ Դաշտի հետ գործողություններ կարելի է կատարել նաև *Datasheet* ներդիրի *Fields&Columns* խմբի համապատասխան գործիքներով:

1. **Ինչպե՞ս են աղյուսակում նոր դաշտ ավելացնում:**
2. **Ինչպե՞ս են աղյուսակում որևէ դաշտ թաքցնում:**
3. **Ինչպե՞ս կարելի է թաքցված դաշտը նորից տեսանելի դարձնել:**

§1.4. ԱՂՅՈՒՍԱԿՈՒՄ ԻՆՖՈՐՄԱՑԻԱՅԻ ՈՐՈՆՈՒՄՆ ՈՒ ՓՈԽԱՐԻՆՈՒՄԸ

Տվյալների հենքում ամենահաճախ կիրառվող գործողություններից մեկը ինֆորմացիայի որոնումն է:

Տվյալների հենքում *անհրաժեշտ ինֆորմացիա որոնելու* համար պետք է.

- կուրսորը տեղադրել փաստաթղթի այն մասում, որտեղից սկսած պետք է որոնումն իրականացնել,
- ընտրել *Home* ներդիրի *Find* խմբի  (*Find*) կոճակը,
- բացված պատուհանում ընտրել *Find* ներդիրը (նկ. 1.15. ա),
- *Look In* դաշտում ընտրել աղյուսակի անվանումը, եթե որոնումը պետք է իրականացնել ողջ աղյուսակում կամ ընտրել աղյուսակի միայն այն դաշտի անվանումը, որտեղ պետք է իրականացվի համապատասխան որոնումը:

Եթե որոնումն իրականացվելու է դաշտում, ապա *Match* դաշտում պետք է ընտրել հետևյալ տարբերակներից անհրաժեշտը.

- *Any Part of Field* – որոնումն իրականացնել դաշտի որևէ մասում,
- *Whole Field* – որոնումն իրականացնել ամբողջ դաշտում,
- *Start of Field* – որոնումն իրականացնել նշված դաշտի սկզբից:

Եթե պետք է որոնել գրառումների մեջ, ապա *Search* դաշտում անհրաժեշտ է ընտրել հետևյալ տարբերակներից մեկը.

- *Up* – որոնել աղյուսակի ընթացիկ գրառումից վերև ընկած մասում,
- *Down* – որոնել աղյուսակի ընթացիկ գրառումից ներքև ընկած մասում,
- *All* – որոնել բոլոր գրառումներում:

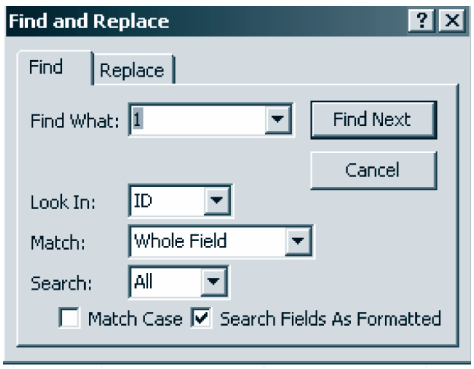
Այժմ որոնման գործընթացը սկսելու համար մնում է տալ փնտրվող ինֆորմացիան, որը պետք է ներմուծել *Find What* դաշտ և սեղմել *Find Next* կոճակը (այսպիսով, համակարգչին «ստիպում» ենք հաջորդաբար ցույց տալ գտածը):

Եթե որոնվածն արդեն գտնվել է կամ որևէ այլ պատճառով հետագա որոնումը նպատակահարմար չէ, ապա որոնման գործընթացն ընդհատելու համար պետք է սեղմել *Cancel* կոճակը:

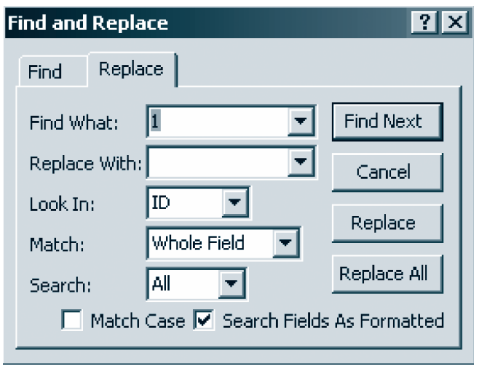
Հենքային տվյալների ղեկավարման *Microsoft Access* համակարգում *ինֆորմացիա որոնելու և այն այլ ինֆորմացիայով փոխարինելու* համար անհրաժեշտ է.

- կուրսորը տեղադրել փաստաթղթի այն մասում, որտեղից սկսած անհրաժեշտ է իրականացնել որոնման ու փոխարինման գործընթացը,
- ընտրել *Home* ներդիրի *Find* խմբի  (*Replace*) հրամանը,
- բացված պատուհանում ընտրել *Replace* ներդիրը (նկ. 1.15. բ),
- բացված պատուհանի *Find What* դաշտում ներմուծել որոնվող ինֆորմացիան, իսկ *Replace With* դաշտում այն ինֆորմացիան, որը պետք է փոխարինի հինին,
- մնացած դաշտերը լրացնել այնպես, ինչպես վերը նկարագրված ինֆորմացիա որոնելու գործընթացում,
- եթե որոնվող ինֆորմացիան աղյուսակում ամենուրեք պետք է փոխարինվի նորով, ապա անհրաժեշտ է սեղմել *Replace All* կոճակը, իսկ եթե անհրաժեշտ է փոխարինել միայն հերթական հանդիպածը, ապա՝ *Replace* կոճակը:

Անհրաժեշտության դեպքում որոնման և փոխարինման գործընթացը կարելի է ընդհատել *Cancel* կոճակով:



ա)



բ)

Նկ. 1.15. Որոնման և փոխարինման պատուհան

ա) Find ներդիր, բ) Replace ներդիր

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ Աղյուսակում ինֆորմացիա կարելի է որոնել նաև՝ օգտվելով դաշտին առնչվող ենթատեքստային մենյուի (նկ. 1.13.) համապատասխան հրամանից:

1. **Ինչպե՞ս են աղյուսակի դաշտում տվյալները կարգավորում աճման կարգով:**
2. **Ինչպե՞ս կարելի է որոնում իրականացնել աղյուսակի որևէ ընթացիկ գրառումից ներքև տեղաբաշխված գրառումների մեջ:**
3. **Ինչպե՞ս պետք է վարվել, եթե աղյուսակի մեջ ամենուրեք մի որոշակի ինֆորմացիա անհրաժեշտ է մեկ այլով փոխարինել:**

Լաբորատոր աշխատանք 1.2
Աշխատանք աղյուսակների հետ

Աշխատանքի ընթացքում Հայաստանի պետական ճարտարագիտական համալսարանի (Պոլիտեխնիկ) *2012* թվականի ընդունելության մասին ստորև բերված ինֆորմացիան նախ կներկայացնենք աղյուսակի տեսքով.

Շիֆր	Ֆակուլտետ	Անվճար տեղերի քանակը	Վճարովի տեղերի քանակը
003071	Քիմիական տեխնոլոգիաների և բնապահպանական ճարտարագիտության	15	90
003073	Էլեկտրատեխնիկական	18	55
003072	Էներգետիկական	37	144
003075	Մեքենաշինության	17	55
003077	Ռադիոտեխնիկայի և կապի համակարգերի	24	95
003078	Կիբեռնետիկայի	45	130
003080	Քոմփյութերային համակարգերի և ինֆորմատիկայի	44	203
003079	Կիրառական մաթեմատիկայի	6	24
003081	Ընդերքաբանության և մետալուրգիայի	22	84
003076	Տրանսպորտային համակարգերի	12	95
003074	Մեխանիկամեքենագիտական	9	62

Ներմուծված տվյալների հենքում այնուհետև պետք է.

ա) *Ֆակուլտետ* դաշտից հետո ավելացնել *Ընդհանուր տեղերի քանակը* դաշտը, որը պարունակելու է ֆակուլտետի *Անվճար տեղերի քանակը* և *Վճարովի տեղերի քանակը* դաշտերի գումարային միավորները,

բ) թաքցնել *Անվճար տեղերի քանակը* և *Վճարովի տեղերի քանակը* դաշտերը,

գ) *ընդհանուր տեղերի քանակը* դաշտը կարգավորել ըստ նվազման,

դ) թաքցված դաշտերը դարձնել տեսանելի,

ե) որոնել *003080* շիֆր ունեցող ֆակուլտետը:

Աշխատանքն իրականացնելու համար կատարենք հետևյալ գործողությունները.

1. Հաջորդաբար կատարելով *Start, All Programs, Microsoft Office, Microsoft Office Access 2007* քայլերը՝ *Access 2007*-ի միջավայր մտեք:
2. Բացված պատուհանում ընտրեք *New Blank Database* բաժնի *Blank Database* կոճակը:
3. Ակտիվացած տիրույթի *File Name* դաշտում ներմուծեք ստեղծվող տվյալների հենքի անվանումը՝ *Lab_1_2_**, որտեղ *-ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:
4.  կոճակով բացված պատուհանում ընտրեք տվյալ դասարանին հատկացված թղթապանակը:
5. Սեղմեք *Create* կոճակը:
6. *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքով բերված ցուցակից ընտրեք  (*Design View*) հրամանը:
7. Ընդունեք աղյուսակի առաջարկվող *Table1* անվանումն ու սեղմեք *OK* կոճակը:
8. Ընտրեք *Arial Armenian Unicode* տառատեսակը:
9. Կոնստրուկտորի պատուհանի առաջին տողի *Field Name* դաշտ ներմուծեք *Շիֆր* բառը: Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
10. Երկրորդ տողի *Field Name* դաշտ ներմուծեք *Ֆակուլտետ* բառը, իսկ *Data Type* դաշտում ընտրեք *Text* տիպը:

11. Երրորդ տողի *Field Name* դաշտ ներմուծեք *Անվճար փեղերի քանակը* տեքստը:
12. Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
13. Չորրորդ տողի *Field Name* դաշտում ներմուծեք *Վճարովի փեղերի քանակը* տեքստը: Մկնիկի ցուցիչը տեղադրեք այդ տողի *Data Type* դաշտի աջ մասի սլաքի վրա, սեղմեք ձախ սեղմակն ու առաջարկվող ցուցակից ընտրեք *Number* տիպը:
14. Աղյուսակի կառուցվածքը ստեղծելուց հետո պատուհանի ղեկավարման սեղմակով փակեք պատուհանը:
15. Աղյուսակի պահպանման մասին տրված հարցմանը պատասխանեք *Yes*:
16. Համաձայնեք առաջարկվող *Table1* անվանմանն ու *OK* կոճակը սեղմելով համոզվեք, որ պատուհանի անցումների տիրույթում *Table1* անվանումով աղյուսակ առաջացավ:
17. Մկնիկի ձախ սեղմակի կրկնակի սեղմումով բացեք *Table1* աղյուսակը:
18. Ստեղծված աղյուսակի բոլոր դաշտերը լրացրեք ըստ վերը բերված աղյուսակի:
19. Մկնիկի ցուցիչը տեղադրեք *Անվճար փեղերի քանակը* դաշտի անվան վրա ու սեղմեք աջ սեղմակը: Բացված ենթատեքստային մենյուի *Insert Column* հրամանով այդ դաշտից առաջ նոր դաշտ ստեղծեք:
20. *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքով բերված ցուցակից ընտրեք  (*Design View*) հրամանն ու բացված պատուհանում ավելացված դաշտի *Field1* անվանման փոխարեն ներմուծեք *Ընդհանուր փեղերի քանակը* անվանումը, իսկ *Data Type* դաշտում ընտրեք *Number* տիպը:
21. *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքով բերված ցուցակից ընտրեք  (*Datasheet View*) հրամանն ու աղյուսակում կատարված փոփոխությունները պահպանելու վերաբերյալ հարցմանը պատասխանեք՝ սեղմելով *Yes* կոճակը:
22. *Ընդհանուր փեղերի քանակը* դաշտում ներմուծեք *Անվճար փեղերի քանակը* և *Վճարովի փեղերի քանակը* դաշտերի գումարային միավորները:

23. Մկնիկի ցուցիչով նշեք *Անվճար տեղերի քանակը* և *Վճարովի տեղերի քանակը* դաշտերն ու մկնիկի աջ սեղմակով բացված ենթատեքստային մենյուի *Hide Columns* հրամանով թաքցրեք դրանք:

24. Մկնիկի ցուցիչը տեղադրեք *Ընդհանուր տեղերի քանակը* դաշտի անվան վրա ու մկնիկի աջ սեղմակով բացված ենթատեքստային մենյուի  գործիքի օգնությամբ այդ դաշտի պարունակությունը կարգավորեք ըստ նվազման:

25. Մկնիկի ցուցիչը տեղադրեք աղյուսակի դաշտերից որևէ մեկի անվան վրա ու աջ սեղմակով բացված ենթատեքստային մենյուից ընտրեք *Unhide Columns* հրամանը:

26. Բացված ցուցակում թաքցված *Անվճար տեղերի քանակը* և *Վճարովի տեղերի քանակը* դաշտերը տեսանելի դարձնելու նպատակով համանուն դաշտերի դիմաց նշում կատարեք ու սեղմեք *Close* կոճակը:

27. Որոնեք աղյուսակում առկա *003080* շիֆրը: Այդ նպատակով.

ա) ընտրեք *Home* ներդիրի *Find* խմբի  (*Find*) հրամանը, ապա *Find* ներդիր պատուհանը,

բ) ողջ աղյուսակի մեջ որոնումն իրականացնելու համար *Look In* դաշտում ընտրեք աղյուսակի *Table1* անվանումը,

գ) *Find What* դաշտում ներմուծեք *003080* թիվն ու սեղմեք *Find Next* կոճակը:

28. Ընտրեք պատուհանի ղեկավարման  սեղմակն ու աղյուսակի պահպանման մասին տրված հարցմանը պատասխանեք *Yes*:

29. Ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման սեղմակից:

§1.5. ՏՎՅԱԼՆԵՐԻ ԲԱԶՄԱՂՅՈՒՍԱԿ ՀԵՆՔԵՐ

Դիտարկենք հետևյալ օրինակը: Դիցուք, անհրաժեշտ է մարզային օլիմպիադայի արդյունքների պահպանման տվյալների հենք ստեղծել (աղյուսակ 1.1), որտեղ, մրցանակակիրների տվյալներից բացի, անհրաժեշտ է տեղեկություններ պահպանել նաև ժյուրիի նախագահի և յուրաքանչյուր մասնակցի դպրոցի վերաբերյալ:

Աղյուսակ 1.1

Առարկա	Ժյուրիի նախագահ	ԲՈՒՀ	Դիպլոմ	Մրցանակակիր	Դպրոց	Հասցե	Տնօրեն	Հեռախոս
Ինֆորմատիկա	Նախագահ1	ԲՈՒՀ1	I կարգի	Մրցան.1	Դպրոց 1	Հասցե 1	Տնօրեն 1	հեռ 1
Ինֆորմատիկա	Նախագահ1	ԲՈՒՀ1	II կարգի	Մրցան.2	Դպրոց 2	Հասցե 2	Տնօրեն 2	հեռ 2
Ինֆորմատիկա	Նախագահ1	ԲՈՒՀ1	III կարգի	Մրցան.3	Դպրոց 2	Հասցե 2	Տնօրեն 2	հեռ 2
Մաթեմատիկա	Նախագահ2	ԲՈՒՀ2	I կարգի	Մրցան.4	Դպրոց 1	Հասցե 1	Տնօրեն 1	հեռ 1
Մաթեմատիկա	Նախագահ2	ԲՈՒՀ2	II կարգի	Մրցան.5	Դպրոց 1	Հասցե 1	Տնօրեն 1	հեռ 1
Մաթեմատիկա	Նախագահ2	ԲՈՒՀ2	III կարգի	Մրցան.6	Դպրոց 3	Հասցե 3	Տնօրեն 3	հեռ 3
Ֆիզիկա	Նախագահ3	ԲՈՒՀ3	I կարգի	Մրցան.7	Դպրոց 4	Հասցե 4	Տնօրեն 4	հեռ 4
Ֆիզիկա	Նախագահ3	ԲՈՒՀ3	II կարգի	Մրցան.8	Դպրոց 3	Հասցե 3	Տնօրեն 3	հեռ 3
Ֆիզիկա	Նախագահ3	ԲՈՒՀ3	III կարգի	Մրցան.9	Դպրոց 5	Հասցե 5	Տնօրեն 5	հեռ 5

Ինչպես տեսնում եք, ժյուրիի նախագահի և օլիմպիադայի մասնակցի դպրոցի վերաբերյալ տվյալները ծանրաբեռնում են աղյուսակը, առավել ևս, որ դրանք որոշ գրառումներում կրկնվում են: Նման դեպքերում ողջ տվյալները միևնույն աղյուսակում պահպանելը նպատակահարմար չէ: Փորձենք պահանջվող տվյալների հենքը մասնատել երեք տարբեր աղյուսակների (աղյուսակ 1.2, 1.3 և 1.4): Ինչպես տեսնում եք, աղյուսակ 1.2-ը ժյուրիի նախագահի և դպրոցի վերաբերյալ միայն մեկական դաշտ է պարունակում՝ *Նախագահի կոդը* և *Դպրոցի կոդը*. այս առումով մնացած ինֆորմացիան համապատասխանաբար 1.3 ու 1.4 աղյուսակներ է տեղափոխվել:

Ստեղծված երեք աղյուսակների միասնականությունն ապահովելու նպատակով անհրաժեշտ է դրանց միջև *կապ* ստեղծել:

Աղյուսակների միջև կարող են գործել *մեկը-մեկին, մեկը-շատին* և *շատը-շատին* կապերը:

Մեկը-մեկին (*1 – 1*) կապի դեպքում (նկ. 1.16. ա) *Աղյուսակ 1*-ի յուրաքանչյուր գրառմանը կարող է *Աղյուսակ 2*-ի միայն մեկ գրառում համապատասխանել և հակառակը՝ *Աղյուսակ 2*-ի յուրաքանչյուր գրառմանը կարող է *Աղյուսակ 1*-ի միայն մեկ գրառում համապատասխանել: Օրինակ՝ այսպիսի կապով կարելի է նկարագրել հետևյալ հարաբերությունը՝ յուրաքանչյուր մարդ

ունի միայն մեկ ծննդյան վկայական և յուրաքանչյուր ծննդյան վկայական պատկանում է միայն մեկ մարդու:

Արտաքին բանալի — Նախագահի կոդը

Արտաքին բանալի — Դպրոցի կոդը

Աղյուսակ 1.2

N	Առարկա	Նախագահի կոդը	Դիպլոմ	Մրցանակակիր	Դպրոցի կոդը
1.	Ինֆորմատիկա	N1	Առաջին կարգի	Մրցանակակիր 1	D1
2.	Ինֆորմատիկա	N1	Երկրորդ կարգի	Մրցանակակիր 2	D2
3.	Ինֆորմատիկա	N1	Երրորդ կարգի	Մրցանակակիր 3	D2
4.	Մաթեմատիկա	N2	Առաջին կարգի	Մրցանակակիր 4	D1
5.	Մաթեմատիկա	N2	Երկրորդ կարգի	Մրցանակակիր 5	D1
6.	Մաթեմատիկա	N2	Երրորդ կարգի	Մրցանակակիր 6	D3
7.	Ֆիզիկա	N3	Առաջին կարգի	Մրցանակակիր 7	D4
8.	Ֆիզիկա	N3	Երկրորդ կարգի	Մրցանակակիր 8	D3
9.	Ֆիզիկա	N3	Երրորդ կարգի	Մրցանակակիր 9	D5

Առաջնային բանալի

Աղյուսակ 1.3

Դպրոցի կոդը	Դպրոց	Հասցե	Տնօրեն	Հեռախոս
D1	Դպրոց 1	Հասցե 1	Տնօրեն 1	հեռ 1
D2	Դպրոց 2	Հասցե 2	Տնօրեն 2	հեռ 2
D3	Դպրոց 3	Հասցե 3	Տնօրեն 3	հեռ 3
D4	Դպրոց 4	Հասցե 4	Տնօրեն 4	հեռ 4
D5	Դպրոց 5	Հասցե 5	Տնօրեն 5	հեռ 5

Առաջնային բանալի

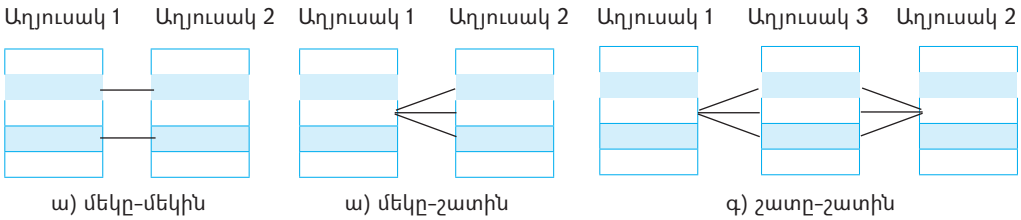
Աղյուսակ 1.4

Նախագահի կոդը	Ժյուրիի նախագահ	ԲՈՒՀ
N1	Նախագահ1	ԲՈՒՀ1
N2	Նախագահ2	ԲՈՒՀ2
N3	Նախագահ3	ԲՈՒՀ3

Մեկը-շատին (*1 - ∞*) կապի դեպքում (նկ. 1.16. բ) *Աղյուսակ 1*-ի յուրաքանչյուր գրառմանը կարող են *Աղյուսակ 2*-ի մի քանի գրառումներ համապատասխանել, իսկ *Աղյուսակ 2*-ի ոչ մի գրառում *Աղյուսակ 1*-ում չի կարող մեկից ավելի համապատասխան գրառում ունենալ: Նման կապ կարելի է հաստատել, օրինակ, այն աղյուսակների միջև, որոնք տվյալներ են նեռառում դպրոցի դասարանների և աշակերտների մասին. ամեն դասարան կարող է բազմաթիվ աշակերտներ ունենալ, բայց յուրաքանչյուր աշակերտ

միայն մի դասարանում է սովորում): Վերը բերված օրինակում աղյուսակ 1.3-ը 1.2-ի հետ կապված է *մեկը-շատին* կապով, որովհետև աղյուսակ 1.2-ում մի շարք դպրոցներ մի քանի անգամ են հանդես գալիս: Աղյուսակ 1.4-ը 1.2-ի հետ ևս *մեկը-շատին* կապով է կապված, քանի որ աղյուսակ 1.2-ում միևնույն ժյուրիի նախագահը ներառվել է երեք անգամ:

Եթե *Աղյուսակ 1*-ի գրառումներին կարող են *Աղյուսակ 2*-ի մի քանի գրառումներ համապատասխանել և *Աղյուսակ 2*-ի գրառումներին էլ *Աղյուսակ 1*-ի մի քանի գրառումներ համապատասխանել, ապա այդպիսի կապն անվանում են *շատը-շատին* ($\infty - \infty$): Երկու աղյուսակների միջև նման *կապ* հնարավոր է իրականացնել *միայն այլ աղյուսակների միջոցով* (նկ. 1.16. գ): *Շատը-շատին* կապով կարելի է նկարագրել, օրինակ, հետևյալ հարաբերությունը՝ բազմաթիվ ուսուցիչներ դասավանդում են բազմաթիվ դասարաններում:



Նկ. 1.16. Աղյուսակների կապի տիպերը

Մեկը-շատին կապի դեպքում աղյուսակը, որի գրառումները կապում մասնակցում են եզակի թվով, համարվում է *գլխավոր* (*առաջնային*), իսկ աղյուսակը, որի մի քանի գրառումներ են ներկա կապում՝ *ենթակա* (*երկրորդային*): Վերը բերված 1.3 ու 1.4 աղյուսակները գլխավոր են աղյուսակ 1.2-ի նկատմամբ, իսկ 1.2-ը՝ ենթակա աղյուսակ 1.3-ի ու 1.4-ի նկատմամբ:

Յուրաքանչյուր աղյուսակ պետք է առնվազն մեկ *բանալի դաշտ* պարունակի, որը եզակի է այդ աղյուսակի յուրաքանչյուր գրառման համար: Բանալի դաշտը թույլատրում է աղյուսակի յուրաքանչյուր գրառում միարժեքորեն նույնականացնել:

> *Դաշտը,* **որի պարունակությամբ աղյուսակի համապատասխան գրառումը միարժեքորեն նույնականացվում է, անվանում են** *բանալի դաշտ*:

Աղյուսակների միջև կապ հաստատելու համար գլխավոր աղյուսակից *առաջնային բանալու* տեղակայման համար ենթակա աղյուսակում պետք է համապատասխան դաշտ նախատեսված լինի, որի արժեքն անվանում են *արտաքին բանալի*:

Աղյուսակների միջև կապ հաստատելիս գլխավոր աղյուսակի *առաջնային բանալի* դաշտի հետ կապվում է ենթակա աղյուսակի նույնանուն *արտաքին բանալի* դաշտը: Բերված օրինակում աղյուսակ 1.3-ի *Դպրոցի կոդը* և աղյուսակ 1.4-ի *Նախագահի կոդը* անվանումներով դաշտերն առաջնային բանալիներ են և աղյուսակ 1.2-ում նույնանուն արտաքին բանալի հանդիսացող դաշտեր ունեն:

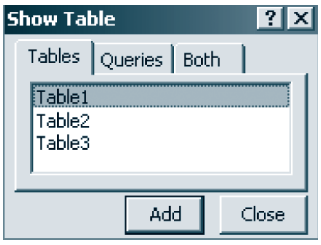
Կապված աղյուսակները **տվյալների միասնական հենք են հանդիսանում, որոնց հիման վրա** *հարցումներ*, *ձևեր* **և** *հաշվետվություններ* **կարելի է կազմակերպել:**

Աղյուսակում բանալի դաշտն ընտրելու համար անհրաժեշտ է.

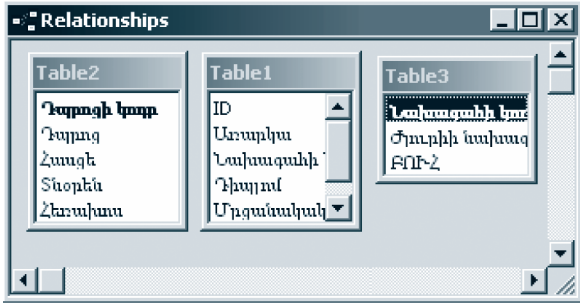
- աղյուսակի կառուցվածքը նախագծելու պատուհանում ընտրել համապատասխան դաշտը,
- ընտրել *Design* ներդիրի *Tools* խմբի կամ մկնիկի աջ սեղմակով բացված մենյուի *Primary Key* կոճակը:

Աղյուսակների միջև կապ հաստատելու համար անհրաժեշտ է.

- ստեղծել անհրաժեշտ աղյուսակները՝ համապատասխան բանալի դաշտերով,
- ընտրել *Database Tools* ներդիրի  (*Relationships*) կոճակը,
- ընտրել այն աղյուսակները, որոնց միջև անհրաժեշտ է կապ հաստատել. բացված *Show Table* պատուհանում (նկ.1.17.) դրա համար պետք է նշել անհրաժեշտ աղյուսակներն ու սեղմել *Add* կոճակը,



Նկ. 1.17. Աղյուսակների ընտրման պատուհան

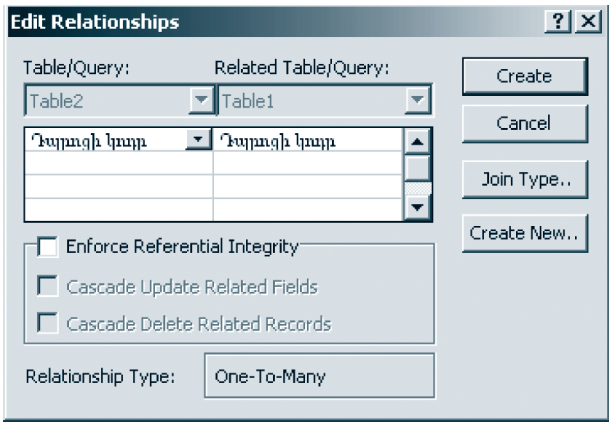


Նկ. 1.18. Relationships պատուհան

- *Close* կոճակով փակել *Show Table* պատուհանը,
- ընտրել *Relationships* պատուհանում (նկ. 1.18.) բերված գլխավոր աղյուսակի բանալի դաշտը (այն արտահայտված է գույնի մգացմամբ) ու

մկնիկի օգնությամբ այն ենթակա աղյուսակի նույնանուն դաշտ տեղափոխել,

- տվյալների ամբողջականությունն ապահովելու նպատակով բացված երկխոսային վահանակից (նկ. 1.19.) ընտրել *Enforce Referential Integrity* դաշտը, ապա *Cascade Update Related Fields* (կապված դաշտերի կասկադային թարմացում) և *Cascade Delete Related Records* (կապված գրառումների կասկադային հեռացում) դաշտերը,
- ընտրել *Create* կոճակը:



Նկ. 1.19. Կապի հաստատման վահանակ

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ Իրար հետ կապված աղյուսակներ պարունակող տվյալների հենքերն անվանում են *ռելյացիոն*:

1. Ինչ նպատակով են մի քանի դաշտեր պարունակող աղյուսակները բաժանում իրար հետ կապված մի քանի աղյուսակների:
2. Ինչի՞ համար է բանալի դաշտը:
3. Աղյուսակների միջև հնարավոր ինչպիսի՞ կապեր գիտեք:
4. Ո՞ր աղյուսակն են անվանում գլխավոր, ո՞րը՝ ենթակա:
5. Ո՞ր բանալին են անվանում առաջնային, ո՞րը՝ արտաքին:
6. Աղյուսակում ինչպե՞ս ընտրել բանալի դաշտը:

Լաբորատոր աշխատանք 1.3
Կապված աղյուսակների ստեղծում

1. Հաջորդաբար կատարելով *Start, All Programs, Microsoft Office, Microsoft Office Access 2007* քայլերը՝ *Access 2007*-ի միջավայր մտեք:
2. Բացված պատուհանում ընտրեք *New Blank Database* բաժնի *Blank Database* կոճակը:
3. Ակտիվացած տիրույթի *File Name* դաշտում ներմուծեք ստեղծվող տվյալների հենքի *Lab_1_3_** անվանումը, որտեղ *–ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:
4. Ընտրեք կոճակը, ապա բացված պատուհանում ընտրեք տվյալ դասարանին հատկացված թղթապանակը:
5. Սեղմեք *Create* կոճակը:
6. Ընտրեք *Arial Armenian Unicode* տառատեսակը:
7. Ընտրեք *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքը, ապա բերված ցուցակից՝ *Design View* հրամանը:
8. Ընդունեք աղյուսակը պահպանելու համար առաջարկվող *Table1* անվանումն ու սեղմեք *OK* կոճակը:
9. Ստորև բերված կառուցվածքով աղյուսակ ստեղծեք.

Table1

Field Name	Data Type
ID	AutoNumber
Առարկա	Text
Նախագահի կողը	Text
Դիպլոմ	Text
Մրցանակակիր	Text
Դպրոցի կողը	Text

10. Աղյուսակի *ID* դաշտը դարձրեք բանալի: Դրա համար ընտրեք *ID* դաշտը, սեղմեք մկնիկի աջ սեղմակն ու ընտրեք *Primary Key* հրամանը:
11. Աղյուսակի կառուցվածքը ստեղծելուց հետո սեղմակով փակեք պատուհանն ու աղյուսակի պահպանման մասին տրված հարցմանը պատասխանեք *Yes*:

12. Մկնիկի ձախ սեղմակի կրկնակի սեղմումով բացեք *Table1* աղյուսակը:
13. Ստորև բերված աղյուսակի օրինակով այն լրացրեք.

Table1

ID	Առարկա	Նախագահի կոդը	Դիպլոմ	Մրցանակակիր	Դպրոցի կոդը
1	Ինֆորմատիկա	N1	I կարգի	Մրցանակակիր1	D1
2	Ինֆորմատիկա	N1	II կարգի	Մրցանակակիր2	D2
3	Ինֆորմատիկա	N1	III կարգի	Մրցանակակիր3	D2
4	Մաթեմատիկա	N2	I կարգի	Մրցանակակիր4	D1
5	Մաթեմատիկա	N2	II կարգի	Մրցանակակիր5	D1
6	Մաթեմատիկա	N2	III կարգի	Մրցանակակիր6	D3
7	Ֆիզիկա	N3	I կարգի	Մրցանակակիր7	D4
8	Ֆիզիկա	N3	II կարգի	Մրցանակակիր8	D3
9	Ֆիզիկա	N3	III կարգի	Մրցանակակիր9	D5

14. Ընտրեք պատուհանի ղեկավարման  սեղմակն ու աղյուսակի պահպանման վերաբերյալ տրված հարցմանը պատասխանեք *Yes*:
15. Նման ձևով ստեղծեք *Դպրոցի կոդը* բանալի դաշտով ստորև բերված *Table2* աղյուսակը:

Table2

Դպրոցի կոդը	Դպրոց	Հասցե	Տնօրեն	Հեռախոս
D1	Դպրոց1	Հասցե1	Տնօրեն1	Հեռախոս1
D2	Դպրոց2	Հասցե2	Տնօրեն2	Հեռախոս2
D3	Դպրոց3	Հասցե3	Տնօրեն3	Հեռախոս3
D4	Դպրոց4	Հասցե4	Տնօրեն4	Հեռախոս4
D5	Դպրոց5	Հասցե5	Տնօրեն5	Հեռախոս5

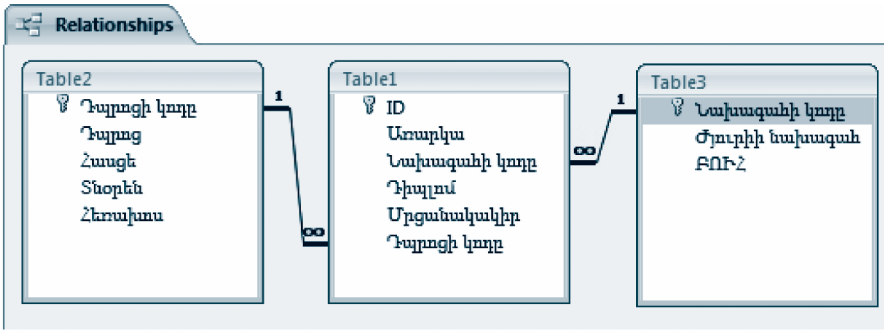
16. Ստեղծեք նաև *Նախագահի կոդը* բանալի դաշտով ստորև բերված *Table3* աղյուսակը:

Table3

Նախագահի կոդը	Ժյուրիի նախագահ	ԲՈՒՀ
N1	Նախագահ1	ԲՈՒՀ1
N2	Նախագահ2	ԲՈՒՀ2
N3	Նախագահ3	ԲՈՒՀ3

17. Ընտրեք *Database Tools* ներդիրի  (*Relationships*) կոճակը:
18. Բացված *Show Table* պատուհանում *Ctrl* ստեղնը սեղմած վիճակում նշեք *Table1*, *Table2* և *Table3* աղյուսակներն ու սեղմեք *Add* կոճակը:
19. *Close* կոճակով փակեք *Show Table* պատուհանը:

20. Ընտրեք *Relationships* պատուհան բերված *Table2* աղյուսակի *Դպրոցի կոդը* բանալի դաշտն ու այն մկնիկի օգնությամբ *Table1* աղյուսակի նույնանուն դաշտ տեղափոխեք:
21. Բացված *Edit Relationships* պատուհանում ընտրեք *Enforce Referential Integrity* դաշտը, ապա *Cascade Update Related Fields* և *Cascade Delete Related Records* դաշտերը:
22. Սեղմեք *Create* կոճակը:
23. Այժմ ընտրեք *Relationships* պատուհան բերված *Table3* աղյուսակի *Նախագահի կոդը* բանալի դաշտն ու մկնիկի օգնությամբ այն *Table1* աղյուսակի նույնանուն դաշտ տեղափոխեք:
24. Ընտրեք *Enforce Referential Integrity* դաշտը, ապա *Cascade Update Related Fields* և *Cascade Delete Related Records* դաշտերը:
25. Սեղմեք *Create* կոճակը: Եթե ամեն ինչ ճիշտ եք կատարել՝ կստանաք հետևյալ կապերը.



26. Աղյուսակների միջև կապերը հաստատելուց հետո պատուհանի ղեկավարման սեղմակով փակեք *Relationships* պատուհանը:
27. Կապված աղյուսակների պահպանման մասին տրված հարցմանը պատասխանեք *Yes*:
28. Ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման սեղմակից:

§1.6. ՀԱՐՑՈՒՄՆԵՐ

Տվյալների հենքի հետ աշխատելիս, որպես կանոն, օգտվողը տվյալների հենքի այս կամ այն աղյուսակում պահպանված ողջ ինֆորմացիան միաժամանակ տեսնելու անհրաժեշտություն չունի: Ընդհակառակը, հաճախ անհրաժեշտ է լինում միաժամանակ ցուցադրել մի քանի աղյուսակների որոշակի պայմանների բավարարող տվյալները:

Access-ում աղյուսակային տվյալները մշակելու հզոր միջոց կա՝ *հարցումը*:

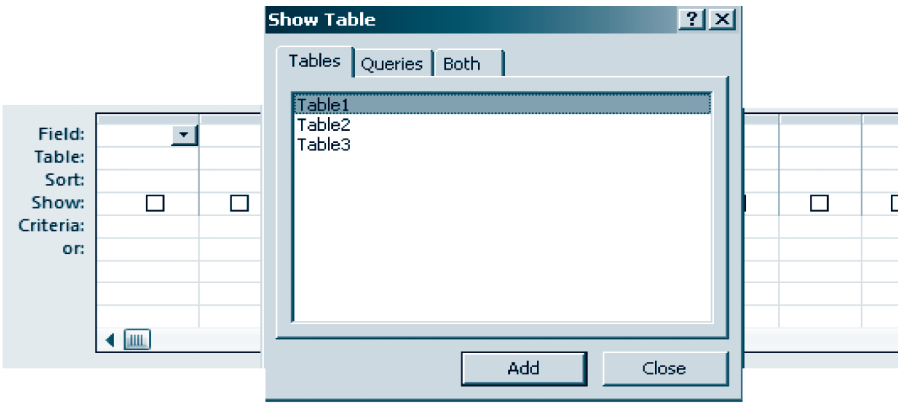
Հարցումը **հնարավորություն է տալիս ցուցադրել աղյուսակի առաջադրված պահանջներին բավարարող տվյալները:**

Հարցումները կարող են լինել *պարզ* և *բաղադրյալ*: *Պարզ հարցումը* մի պայման է պարունակում, իսկ *բաղադրյալ հարցումը*՝ տարբեր դաշտերի համար նախատեսված մի քանի պայմաններ:

Հարցումներ (*Queries*) ստեղծելու երկու եղանակ կա.

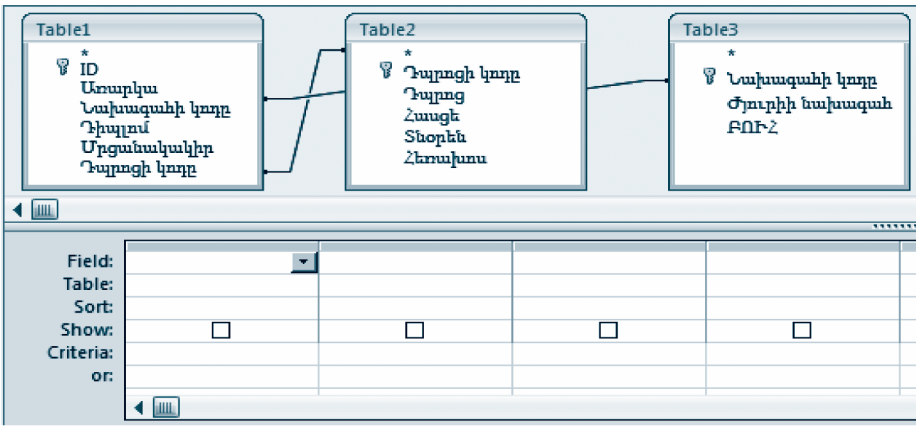
- *Query Design* – *կոնստրուկտորի օգնությամբ,*
- *Query wizard* – *շաբլոնների օգնությամբ:*

Մենք *Access 2003*-ում ուսումնասիրել ենք կոնստրուկտորի օգնությամբ հարցումներ ստեղծելու գործընթացը: Այժմ *Access 2007*-ում այդ գործընթացն ուսումնասիրենք *կապված աղյուսակների* համար, որի դեպքում նախ անհրաժեշտ է *Create* ներդիրի *Other* խմբից ընտրել *Query Design* անվանումով կոճակը: Այնուհետև բացված *Show Table* պատուհանում (նկ. 1.20.) պետք է *Ctrl* ստեղնը սեղմած վիճակում ընտրել անհրաժեշտ աղյուսակներն ու սեղմել *Add* կոճակը:



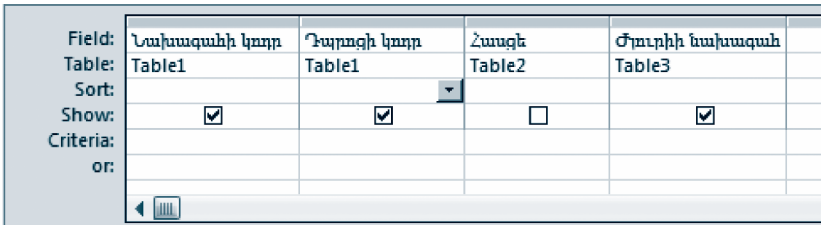
Նկ. 1.20. Հարցումներ ձևակերպելու բլանկ

Այս եղանակով աղյուսակներն ընտրելուց հետո բացվում է *հարցումներ ձևակերպելու բլանկը*, որի վերին մասում բերվում են ընտրված աղյուսակները՝ համապատասխան կապերով (նկ. 1.21.): Բլանկի ստորին մասում երևում է հարցումներ ձևակերպելու բլանկը:



Նկ. 1.21. Ընտրված աղյուսակներն ու հարցումներ ձևակերպելու բլանկը

Հարցումների բլանկը լրացնելուց առաջ պետք է մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարել պատուհանի վերին մասում տեղակայված աղյուսակների այն դաշտերի վրա, որոնց վերաբերյալ պետք է հարցում կատարել: Այդ դեպքում բլանկի *Field* անվամբ տողի բջիջներում կտեղադրվեն աղյուսակի ընտրված դաշտերը, իսկ *Table* անվամբ տողի բջիջներում՝ աղյուսակի անվանումը (նկ. 1.22.):



Նկ. 1.22. Լրացված դաշտերով հարցումներ ձևակերպելու բլանկ

Sort տողը նախատեսված է ընտրված դաշտն ըստ *աճման* կամ *նվազման* կարգավորելու համար: Կարգավորման նպատակով անհրաժեշտ է մկնիկի ցուցիչը տեղադրել համապատասխան բջիջի վրա և այդ բջիջի աջ մասում տեղակայված սլաքով բացված ցուցակից ըստ անհրաժեշտության ընտրել առաջարկվող տարբերակներից որևէ մեկը՝ *Ascending* – կարգավորել ըստ աճման, *Descending* – կարգավորել ըստ նվազման, *not sorted* – տվյալները չկարգավորել:

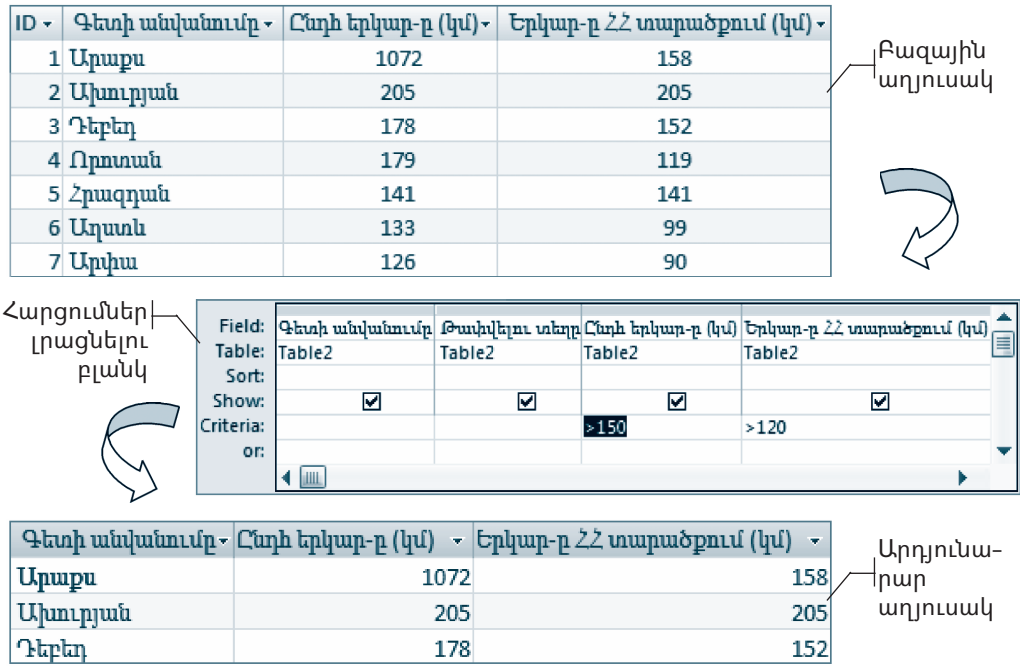
Show տողը նախատեսված է նշված դաշտերը էկրանին ցուցադրելու համար: Եթե դաշտի համապատասխան տողը նշված է (☑), ապա արդյունա-

րար աղյուսակում տվյալ դաշտը կցուցադրվի, իսկ եթե նշված չէ՝ (☐), չի ցուցադրվի, սակայն այդտեղ ներառված տվյալները կարելի է օգտագործել:

Criteria տողի դաշտերում կարելի է նշել այն պայմանները, որոնց համաձայն տվյալ դաշտի արժեքներից պետք է ընտրվեն անհրաժեշտները: *Or* տողը նախատեսված է այդ դաշտերում լրացուցիչ պայմաններ ներմուծելու համար:

Criteria տողում յուրաքանչյուր դաշտի համար կարելի է առանձին պայմաններ նշել. այս դեպքում ընտրվում են այն գրառումները, որոնք միաժամանակ բավարարում են բոլոր դաշտերում նշված պայմաններին: Նկ. 1.23-ում բերված օրինակում Հայաստանի գետերի մասին ինֆորմացիա պարունակող աղյուսակից հարցումների միջոցով ընտրվել են այն գետերը, որոնց ընդհանուր երկարությունը գերազանցում է *150* կմ-ը, և որոնք հանրապետության տարածքում *120* կմ-ից ավելի ձգվող հուն ունեն:

Ստեղծված *հարցումը կարելի է պահպանել* արագ հասանելիության վահանակի կամ *Office* կոճակով բացված մենյուի գործիքով:



ID	Գետի անվանումը	Ընդհ երկար-ը (կմ)	Երկար-ը ՀՀ տարածքում (կմ)
1	Արաքս	1072	158
2	Ախուրյան	205	205
3	Դեբեդ	178	152
4	Ողոտան	179	119
5	Հրազդան	141	141
6	Աղստև	133	99
7	Արփա	126	90

Field:	Գետի անվանումը	Թափվելու տեղը	Ընդհ երկար-ը (կմ)	Երկար-ը ՀՀ տարածքում (կմ)
Table:	Table2	Table2	Table2	Table2
Sort:				
Show:	☑	☑	☑	☑
Criteria:			>150	>120
or:				

Գետի անվանումը	Ընդհ երկար-ը (կմ)	Երկար-ը ՀՀ տարածքում (կմ)
Արաքս	1072	158
Ախուրյան	205	205
Դեբեդ	178	152

Նկ. 1.23. Ընտրության պայմանի աշխատանքի օրինակ

Նոր ստեղծված *հարցումը կարելի է իրագործել Design* ներդիրի *Results* խմբի  (*Run*) գործիքի միջոցով: Հարցումների բլանկը կրկին *կարելի է բացել Home* ներդիրի *Views* խմբի  (*View*) կոճակի ստորին մասում տեղակայված ▼ սլաքով բացվող ցուցակից ընտրելով *Design View* կոճակը: Նախկինում ստեղծված և արդեն պահպանված հարցումը կարելի է իրագործել *Անցումների փիրույթում* համապատասխան օբյեկտի վրա մկնիկի ձախ սեղմակի կրկնակի սեղմումով:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *Criteria* և *or* տողերի ընտրման պայմանները կարող են ներառել «*» և «?» հատուկ նշանակության պայմանանշանները, որտեղ *-ը փոխարինում է ցանկացած քանակությամբ պայմանանշանների, իսկ ?-ը՝ մեկ պայմանանշանի:

1. **Ինչի՞ համար է *Queries* օբյեկտը:**
2. **Ինչպիսի՞ տողերից է բաղկացած հարցումներ ձևակերպելու բլանկը:**
3. **Ինչի՞ համար է հարցումներ ձևակերպելու բլանկի *Sort* տողը:**
4. **Ինչի՞ համար է հարցումներ ձևակերպելու բլանկի *Show* տողը:**
5. **Ի՞նչ են նշում հարցումներ ձևակերպելու բլանկի *Criteria* և *or* տողերում:**

Լաբորատոր աշխատանք 1.4
Կապված աղյուսակներին առնչվող հարցումներ

1. Մտեք տվյալների հենքերի ղեկավարման *Access 2007* համակարգի միջավայր:
2. Բացված պատուհանում ընտրեք *New Blank Database* բաժնի *Blank Database* կոճակը:
3. Ակտիվացած տիրույթի *File Name* դաշտում ներմուծեք ստեղծվող տվյալների հենքի *Lab_1_4_** անվանումը, որտեղ *-ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:

4. Ընտրեք  կոճակը, ապա բացված պատուհանում տվյալ դասարանին հատկացված թղթապանակը:
5. Սեղմեք *Create* կոճակը:
6. Ընտրեք *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքը, ապա բերված ցուցակից՝ *Design View* հրամանը:
7. Աղյուսակը պահպանելու համար առաջարկվող *Table1* անվան փոխարեն ներմուծեք *Arajadimutyun* անունն ու սեղմեք *OK* կոճակը:
8. Ընտրեք *Arial Armenian Unicode* տառատեսակը:
9. Ստեղծեք հետևյալ կառուցվածքով աղյուսակը.
10. Աղյուսակի *ID* դաշտը դարձրեք *բանալի*: Դրա համար ընտրեք *ID* դաշտը, սեղմեք մկնիկի աջ սեղմակն ու ընտրեք *Primary Key* հրամանը:
11. Արագ հասանելիության վահանակի  (*Save*) կոճակով պահպանեք ստեղծված աղյուսակի կառուցվածքը:
12. Ընտրեք *Home* ներդիրի *Views* խմբի *View* կոճակի ստորին մասի ▼ սլաքը, ապա բերված ցուցակից՝ *Datasheet View* հրամանը: Աղյուսակը լրացրեք ստորև բերված տվյալներով.

Arajadimutyun

ID	Առարկայի կոդը	Աշակերտի կոդը	Գնահատականը	Ստուգման ձևը
1	1	1	գեր	քննություն
2	1	2	լավ	քննություն
3	1	3	գեր	քննություն
4	1	4	լավ	քննություն
5	1	5	բավ	քննություն
6	1	6	գեր	քննություն
7	1	7	գեր	քննություն
8	2	1	լավ	քննություն
9	2	2	բավ	քննություն
10	2	3	գեր	քննություն
11	2	4	գեր	քննություն
12	2	5	գեր	քննություն
13	2	6	լավ	քննություն
14	2	7	գեր	քննություն
15	3	1	բավ	քննություն
16	3	2	գեր	քննություն
17	3	3	լավ	քննություն
18	3	4	գեր	
19	3	5	գեր	
20	3	6	բավ	
21	3	7	գեր	

Arajadimutyun

Field Name	Data Type
ID	AutoNumber
Առարկայի կոդը	Number
Աշակերտի կոդը	Number
Գնահատականը	Text
Ստուգման ձևը	Text

13. Ընտրեք *Create* ներդիրի *Tables* խմբի *Table Design* կոճակն ու ստեղծեք *Ararka* անունով աղյուսակ՝ *Առարկայի կոդը* բանալի դաշտով: Աղյուսակը լրացրեք ստորև բերված տվյալներով.

Ararka

Առարկայի կոդը	Առարկան	Ժամերի քանակը
1	Ինֆորմատիկա	34
2	Մաթեմատիկա	68
3	Ֆիզիկա	68

14. Ստեղծեք *Ashakert* անունով աղյուսակ՝ *Աշակերտի կոդը* բանալի դաշտով: Աղյուսակը լրացրեք ստորև բերված տվյալներով.

Ashakert

Աշակերտի կոդը	Դասարանի կոդը	Ազգանունը	Անունը	Սեռը	Ծննդյան թիվը
1	1	Պետոյան	Արմեն	Արական	04.03.95
2	1	Սիմոնյան	Սոնա	Իգական	12.12.96
3	2	Գևորգյան	Կարեն	Արական	25.04.95
4	3	Ավագյան	Աննա	Իգական	24.12.95
5	2	Անտոնյան	Մայիս	Արական	17.07.95
6	3	Սիմոնյան	Նինա	Իգական	24.12.96
7	1	Գևորգյան	Լևոն	Արական	12.08.95

15. Ստեղծեք *Dasaran* անունով աղյուսակ՝ *Դասարանի կոդը* բանալի դաշտով: Աղյուսակը լրացրեք ստորև բերված տվյալներով.

Dasaran

Դասարանի կոդը	Դասարանը	Դասղեկը
1	12ա	Ավագյան Կ.
2	12բ	Սիմոնյան Բ.
3	12գ	Կարապետյան Ն.
*		

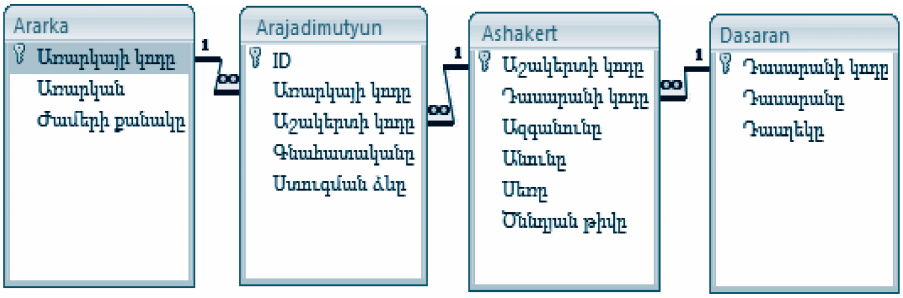
16. Ընտրեք մենյուի տողի *Database Tools* ներդիրի *Show/Hide* խմբի *Relationships* հրամանը: Բացված *Show Table* պատուհանում *CTRL* ստեղնը սեղմած պահելով՝ նշեք ստեղծված *4* աղյուսակներն ու սեղմեք *Add* կոճակը: *Close* կոճակով փակեք *Show Table* պատուհանը:

17. Ընտրեք *Relationships* պատուհան բերված *Ararka* աղյուսակի *Առարկայի կոդը* բանալի դաշտն ու մկնիկի օգնությամբ այն *Arajadimutyun* աղյուսակի նույնանուն դաշտ տեղափոխեք:

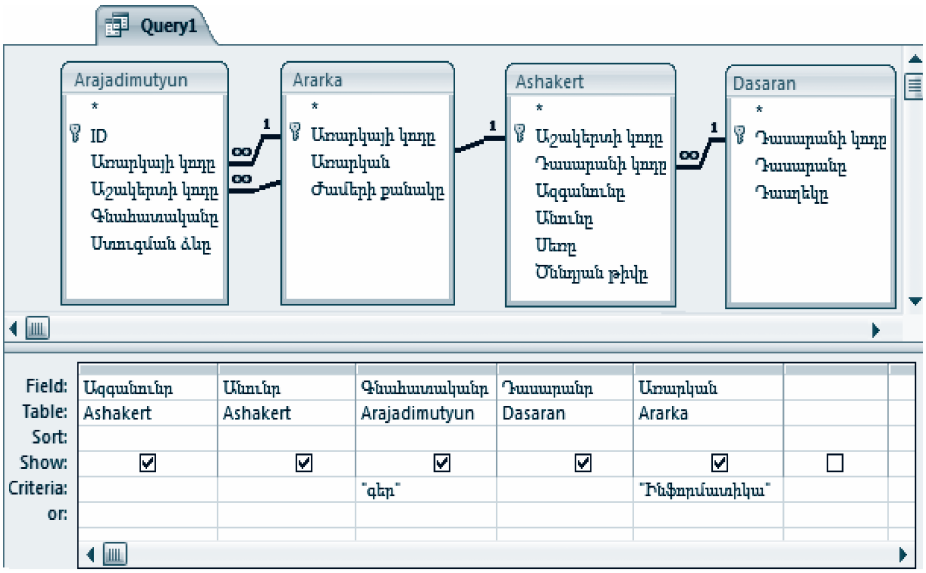
18. Ընտրեք *Enforce Referential Integrity* դաշտը, ապա *Cascade Update Related Fields* և *Cascade Delete Related Records* դաշտերը:

19. Սեղմեք *Create* կոճակը:

20. Նման ձևով, ստորև բերված նկարի համաձայն, աղյուսակների միջև *մեկը-շատերին* տիպի կապ հաստատեք.



21. Անհրաժեշտ կապերը հաստատելուց հետո պատուհանի ղեկավարման սեղմակով փակեք *Relationships* պատուհանը:
22. Արվածը պահպանելու մասին տրված հարցմանը պատասխանեք *Yes*:
23. *Create* ներդիրի *Other* խմբից ընտրեք *Query Design* անվանումով կոճակը:
24. Բացված *Show Table* պատուհանում *Ctrl* ստեղնը սեղմած վիճակում ընտրեք ստեղծված *4* աղյուսակներն ու սեղմեք *Add* կոճակը:
25. *Close* կոճակով փակեք *Show Table* պատուհանը:
26. Մկնիկի ձախ սեղմակով հաջորդաբար կրկնակի սեղմում կատարեք պատուհանի վերին մասում տեղակայված *Ashakert* աղյուսակի *Ազգանուն* և *Անուն*, *Arajadimwutyun* աղյուսակի *Գնահատականը*, *Dasaran* աղյուսակի *Դասարանը* և *Ararka* աղյուսակի *Առարկան* դաշտերի վրա: Այդ դեպքում բլանկի *Field* անվամբ տողի բջիջներում կտեղադրվեն աղյուսակի ընտրված դաշտերը, իսկ *Table* անվամբ տողի բջիջներում՝ աղյուսակի անվանումը:



27. *Criteria* տողի *գնահատականը* դաշտում ներմուծելով *«գեր»*, իսկ *Առարկան* դաշտում *«Ինֆորմատիկա»* հարցումներն ու ընտրելով *Design* ներդիրի *Results* խմբի 

գործիքը՝ արդյունքում տարբեր, բայց իրար հետ կապված աղյուսակներից էկրանին ինֆորմացիա կբերվի այն աշակերտների մասին, որոնք Ինֆորմատիկայից գերազանց գնահատական են ստացել.

Query1

Ազգանունը	Անունը	Գնահատականը	Դասարանը	Առարկան
Պետոյան	Արմեն	գեր	12ա	Ինֆորմատիկա
Գևորգյան	Կարեն	գեր	12բ	Ինֆորմատիկա
Սիմոնյան	Նինա	գեր	12գ	Ինֆորմատիկա
Գևորգյան	Լևոն	գեր	12ա	Ինֆորմատիկա

28. Արագ հասանելիության վահանակի  (*Save*) կոճակով պահպանեք ստեղծված հարցումը:

29. Ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման  սեղմակից:

§1.7. ՁԵՎԵՐ

Մենք արդեն սովորել ենք աշխատել այնպիսի աղյուսակների հետ, որտեղ յուրաքանչյուր գրառում աղյուսակի տող է ներկայացնում: Տվյալների հենքի նման ներկայացումը հնարավորություն է տալիս էկրանին միաժամանակ մի քանի գրառումներ դիտել: Սակայն տվյալների հենքը հաճախ մեծ քանակությամբ դաշտեր է պարունակում, դաշտերն էլ՝ բազմաթիվ պայմանանշաններ, և հնարավոր չի լինում միաժամանակ ողջ ինֆորմացիան էկրանին ամբողջությամբ դիտել: Նման դեպքերում տվյալների ներմուծման և խմբագրման գործընթացներն անհամեմատ դժվարանում են: Առավել բարդ է տվյալներ ներմուծել միաժամանակ մի քանի աղյուսակներում, քանի որ յուրաքանչյուր աղյուսակում տվյալներ ներմուծելուց առաջ ստիպված ենք այդ աղյուսակը բացել:

Բարդություններից խուսափելու համար աղյուսակում տվյալներ ներմուծելու լավագույն տարբերակը *ձևերի* ստեղծումն է, որը թույլատրում է տվյալներ ներմուծել միաժամանակ մի քանի աղյուսակներում, ընդ որում՝ յուրաքանչյուր դաշտի համար հատկացնելով այնքան տեղ, որքան անհրաժեշտ է: Նման ձևերի կիրառմամբ տվյալներ ներմուծելու գործընթացը

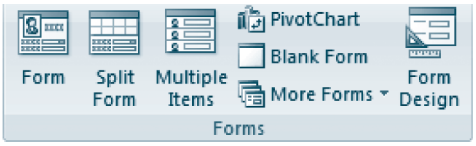
հնարավորինս պարզեցվում է, քանի որ յուրաքանչյուր ձևում նույն պահին էկրանին միայն մեկ գրառում է ցույց տրվում:

Ձևերը ներառում են *կառավարման տարրեր* (տեքստային դաշտեր, կոճակներ, փոխանջատիչներ և այլն) ու *մակագրություններ*, որոնք հեշտացնում են ձևի օգտագործումը և լրացնում դրա արտաքին տեսքը: Հաճախ ձևերին կցված մակագրությունները դաշտերի անվանումներ են, իսկ տեքստային դաշտերը երբեմն թվային տվյալներ են պարունակում:

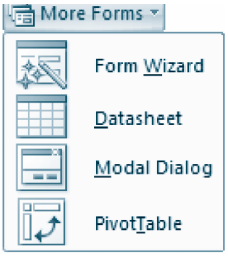
Ձևեր ստեղծելու տարբեր միջոցներ գոյություն ունեն: Մենք ձևեր ստեղծելու գործընթացն ուսումնասիրելու ենք *վարպետի* օգնությամբ:

Վարպետի օգնությամբ ձև սրեղծելու համար նախ անհրաժեշտ է.

- ընտրել *Create* ներդիրի *Forms* խմբի (նկ. 1.24.) *More Forms* կոճակը,
- բացված մենյուից (նկ. 1.25.) ընտրել *Form Wizard* հրամանը:



Նկ. 1.24. Create ներդիրի Forms խումբ

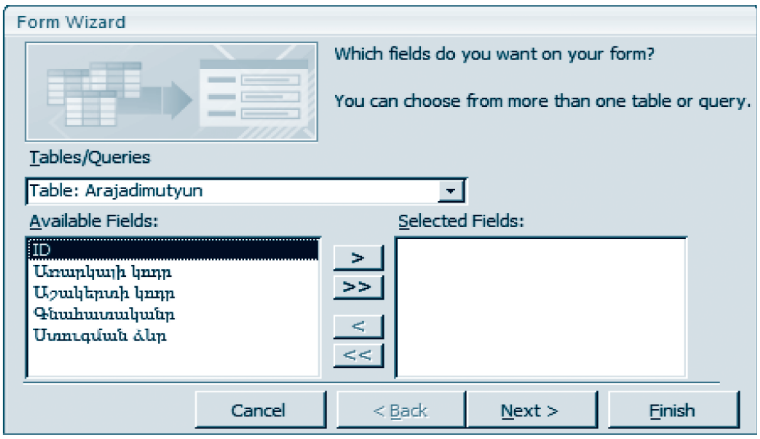


Նկ. 1.25. More Forms մենյու

Ձևերի վարպետի կիրառման *առաջին փուլում պետք է ընտրել աղյուսակի կամ հարցման* այն *դաշտերը*, որոնք պետք է ստեղծվող ձևի համար *աղբյուր* հանդիսանան: Դրա համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել բացված *Form Wizard* պատուհանի *Tables/Queries* դաշտի (նկ. 1.26.) աջ մասի ▼ սլաքի վրա, սեղմել ձախ սեղմակն ու բացված ցուցակից ընտրել այն աղյուսակը կամ հարցումը, որը պետք է ստեղծվող ձևի համար տվյալների աղբյուր հանդիսանա (նկ. 1.26.–ում բերված օրինակում ընտրվել է *Arajadimutyun* անվանումով աղյուսակը),
- *Available Fields* դաշտում նշել ընտրված աղյուսակի կամ հարցման այն դաշտերը, որոնք պետք է ներառել ստեղծվող ձևում, սեղմել *Next* կոճակը (ընտրված դաշտերի անվանումներն այս քայլերի արդյունքում կարտացոլվեն *Selected Fields* դաշտում):

Form Wizard պատուհանի *Available Fields* դաշտում ընտրված աղյուսակի կամ հարցման դաշտերը կարելի է ավելացնել ձևում > և >> կոճակների օգնությամբ (> – ավելացնում է միայն ընտրված դաշտը, >> – ավելացնում են բոլոր դաշտերը):



Նկ. 1.26. Ձևերի վարպետի 1-ին երկխոսական պատուհանը. դաշտերի ընտրություն

Form Wizard պատուհանի *Selected Fields*-ում արդեն ընտրված դաշտերը ստեղծվող ձևից անհրաժեշտության դեպքում կարելի է հանել < և << կոճակների օգնությամբ (< – հեռացնում է միայն ընտրված դաշտը, << – հեռացնում է բոլոր դաշտերը):

Ձևերի վարպետի կիրառման *երկրորդ փուլում պետք է ընտրել ստեղծվող ձևում տվյալներն արտապատկերելու տեսքը*: Դրա համար անհրաժեշտ է.

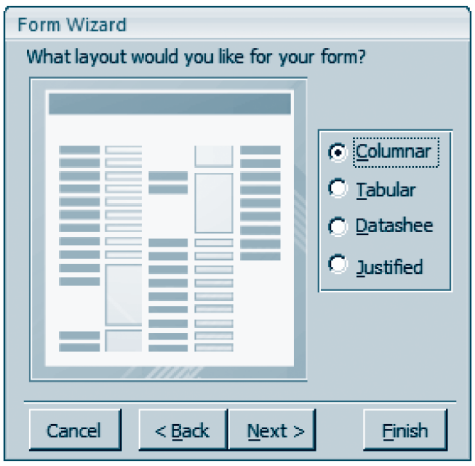
- *Form Wizard* պատուհանից (նկ. 1.27.) ընտրել *Columnar* (սյունակով), *Tabular* (ժապավենային), *Datasheet* (աղյուսակային), *Justified* (հավասարեցված) տարբերակներից անհրաժեշտն ու սեղմել *Next* կոճակը:

Ծանոթանանք ձևում տվյալներ արտապատկերելու հնարավոր տեսքերից երեքին.

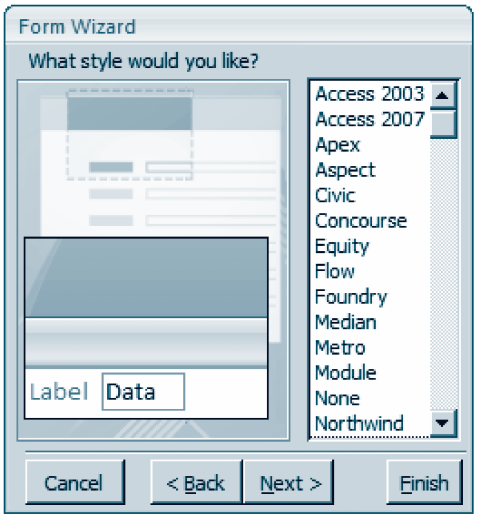
- *Columnar* – սա *սյունակով* կազմակերպված ձև է: Այսպիսի ձևում գրառման դաշտերը, որպես կառավարման տարրեր, տեղադրվում են մեկ կամ մի քանի սյունակներով: Այդ պատճառով սա արագ ստեղծվող ձևերից ամենահավաքն է:
- *Tabular* – սա, այսպես կոչված, *ժապավենային* ձև է: Այս ձևում գրառման դաշտերը, որպես կառավարման տարրեր, տեղադրվում են առանձին մեկ դաշտով: Սա շատ հարմար է մեծաքանակ տվյալների հետ աշխատելիս, որովհետև տվյալներն այս դեպքում տեղաբաշխվում են այնպես, ինչպես սովորական աղյուսակում: Քանի որ յուրաքանչյուր դաշտ ներկայացվում է առանձին կառավարման տարրով, ապա ամեն տարրի համար կարելի է իր ձևաչափը սահմանել, որն էլ այս ձևի առավելությունն է:
- *Datasheet* – սա *աղյուսակի* տեսքով կազմակերպված ձև է: Այս ձևը *Access*-ում ստեղծված սովորական աղյուսակի տեսք ունի:

Ձևերի վարպետի կիրառման *երրորդ փուլում պետք է ընտրել ձևավորման ոճը:* Դրա համար անհրաժեշտ է.

- *Form Wizard* պատուհանից (նկ. 1.28.) ընտրել առաջարկվող ստանդարտ ոճերից անհրաժեշտն ու սեղմել *Next* կոճակը:



Նկ. 1.27. Ձևերի վարպետի 2-րդ երկխոսական պատուհան. արտաքին տեսքի ընտրություն



Նկ. 1.28. Ձևերի վարպետի 3-րդ երկխոսական պատուհան. ձևավորման ոճի ընտրություն

Ձևերի վարպետի կիրառման վերջին՝ *չորրորդ փուլում պետք է պահպանել սպեղծված ձևը*: Դրա համար անհրաժեշտ է.

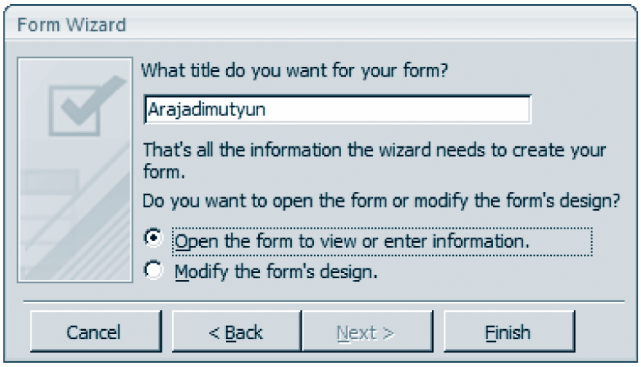
- *Form Wizard* պատուհանում (նկ. 1.29.) ներմուծել ձևի անվանումը կամ համաձայնել համակարգի կողմից առաջարկվող անվանը,
- եթե ձևը ստեղծելուց հետո այն փոփոխելու անհրաժեշտություն է ծագել, ապա պետք է ընտրել *Modify the form's design* (փոխել ձևի կառուցվածքը) փոխանջատիչը, հակառակ դեպքում՝ *Open the form or enter information* (ձևը բացել այն դիտելու կամ տվյալներ ներմուծելու համար) փոխանջատիչը,
- սեղմել *Finish* կոճակը:

Ձևում տվյալներ ներմուծելու համար պետք է.

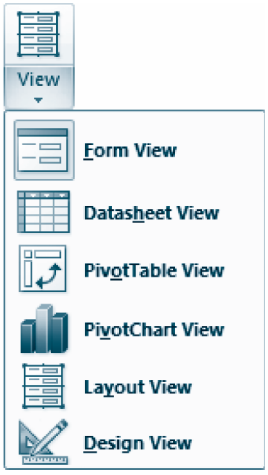
- անցումների տիրույթում անհրաժեշտ ձևի վրա մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարել,
- ձևի դաշտերում առկա գրառումները ջնջելու համար ընտրել *Home* ներդիրի *Records* խմբի New կոճակը,
- ներմուծել դաշտերի տվյալները,
- ներմուծված տվյալները պահպանել արագ հասանելիության վահանակի (*Save*) կոճակով:

Վարպետի օգնությամբ ստեղծված ձևը կոնստրուկտորի միջավայրում խմբագրելու համար անհրաժեշտ է.

- ընտրել *Home* ներդիրի *View* կոճակը,
- բացված պատուհանից (նկ. 1.30.) ընտրել *Design View* հրամանը,
- կատարել անհրաժեշտ խմբագրումը,
- հարկ եղած դեպքում *View* կոճակով բացվող պատուհանի *Form View* հրամանով վերադառնալ ձևի դիտման ռեժիմին:



Նկ. 1.29. Ձևերի վարպետի վերջին երկխոսական պատուհանը. ձևի պահպանում



Նկ. 1.30. Ձևի ռեժիմի ընտրություն

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ Ձևի դաշտերում տվյալներ ներմուծելիս հաջորդ դաշտին կարելի է անցնել *Tab* սեղմակով, իսկ նախորդին՝ *Shift+Tab*-ով:

1. **Ինչ նպատակով են ստեղծվում ձևերը:**
2. **Ստեղծվող ձևերի ինչ տեսքեր գիտեք:**

Լաբորատոր աշխատանք 1.5
Ձևի սրեղծում

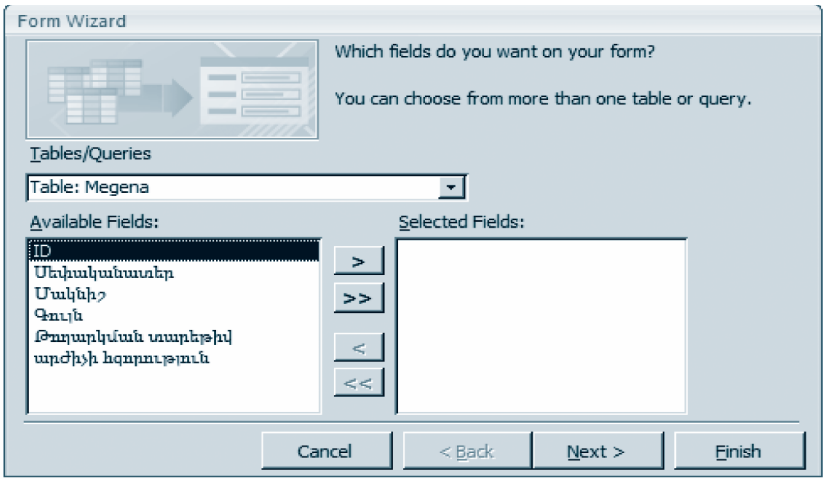
1. Տվյալների հենքերի ղեկավարման *Access 2007* համակարգի միջավայր մտեք:
2. Բացված պատուհանում ընտրեք *New Blank Database* կոճակը:
3. Ակտիվացած տիրույթի *File Name* դաշտում ներմուծեք ստեղծվող տվյալների հենքի *Lab_1_5_** անվանումը, որտեղ *–ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:
4. Ընտրեք կոճակը, ապա բացված պատուհանում՝ տվյալ դասարանին հատկացված թղթապանակը:
5. Սեղմեք *Create* կոճակը:
6. Նոր աղյուսակ ստեղծելու համար ընտրեք *Create* ներդիրի *Tables* խմբի *Table Design* կոճակը:
7. Ընտրեք *Arial Armenian Unicode* տառատեսակը:
8. Ստեղծեք ստորև բերված կառուցվածքով աղյուսակը.

Table1

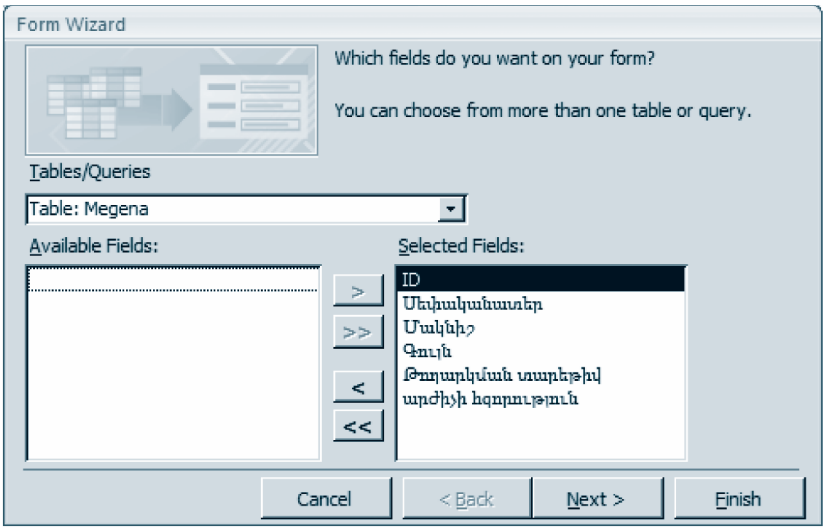
	Field Name	Data Type
🔑	ID	AutoNumber
	Մեխանակատերը	Text
	Մականիշը	Text
	Գույնը	Text
	Տարեթիվը	Number
	Հզորությունը	Number

9. Աղյուսակի *ID* դաշտը դարձրեք բանալի: Դրա համար ընտրեք *ID* դաշտը, սեղմեք մկնիկի աջ սեղմակն ու ընտրեք *Primary Key* հրամանը:
10. Աղյուսակի կառուցվածքը ստեղծելուց հետո սեղմակով փակեք պատուհանը:
11. Աղյուսակը պահպանելու վերաբերյալ տրված հարցմանը պատասխանեք *Yes*:

12. Առաջարկվող անվան փոխարեն ներմուծեք *Meqena* անվանումն ու սեղմելով *OK* կոճակը համոզվեք, որ պատուհանի ձախ վահանակին *Meqena* անվանումով աղյուսակ առաջացավ:
13. Ընտրեք *Create* ներդիրի *Forms* խմբի *More Forms* կոճակը, ապա բացված պատուհանի *Form Vizard* հրամանը:
14. Քանի որ տվյալ հենքում ընդամենը մեկ աղյուսակ ունենք, ապա *Form Wizard* պատուհանի *Tables/Queries* դաշտում բերվել է *Meqena* աղյուսակը՝ *Table:Meqena*:
15. *Available Fields*-ում ներառված բոլոր դաշտերն ավելացրեք ստեղծվող ձևի մեջ՝ դրա համար ընտրելով >> կոճակը:
16. *Selected Fields*-ում ավելացված դաշտերը ստեղծվող ձևից հանելու համար ընտրեք << կոճակը:



17. *Selected Fields*-ում ավելացված դաշտերը ստեղծվող ձևից փորձեք հանել < կոճակի օգնությամբ: Այդ նպատակով *Selected Fields*-ում նշեք անհրաժեշտ դաշտն ու ընտրեք < կոճակը:
18. Այժմ փորձեք > կոճակի օգնությամբ *Available Fields*-ում ներառված դաշտերն ավելացնել ստեղծվող ձևի մեջ:
19. Բացված պատուհանում ընտրեք ստեղծվող ձևի *Tabular* տեսքն ու սեղմեք *Next* կոճակը:
20. Բացված պատուհանում հաջորդաբար փորձեք այստեղ առաջարկվող ստանդարտ ոճերից յուրաքանչյուրը: Վերջում ընտրեք *Northwind* ոճն ու սեղմեք *Next* կոճակը:



21. Բացված պատուհանում ներմուծեք ստեղծվող ձևի *Meqena1* անվանումը: Այժմ ստեղծվող ձևը փոփոխելու հնարավորություն ունենալու համար ընտրեք *Modify the form's design* փոխանջատիչը:
22. Սեղմելով *Finish* կոճակը՝ հնարավորություն ստացեք փոփոխելու ստեղծված ձևի դաշտերի չափերը:
23. Մկնիկի ցուցիչով հաջորդաբար ընտրեք ստեղծված դաշտերից յուրաքանչյուրն ու մկնիկի ցուցիչը տեղադրեք դաշտի եզրագծի վրա և երբ այն կստանա երկկողմ սլաքի տեսք՝ մկնիկի տեղաշարժմամբ փոփոխեք դաշտի չափերն ու ստացեք հետևյալ տեսքի ձև.

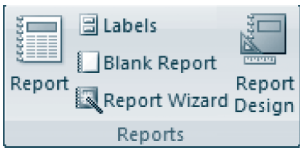
24. Ձեր ցանկությամբ համապատասխան տվյալներ ներմուծեք և ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման  սեղմակից:

§1.8. ՀԱՇՎԵՏՎՈՒԹՅՈՒՆՆԵՐԻ ՍՏԵՂԾՈՒՄ

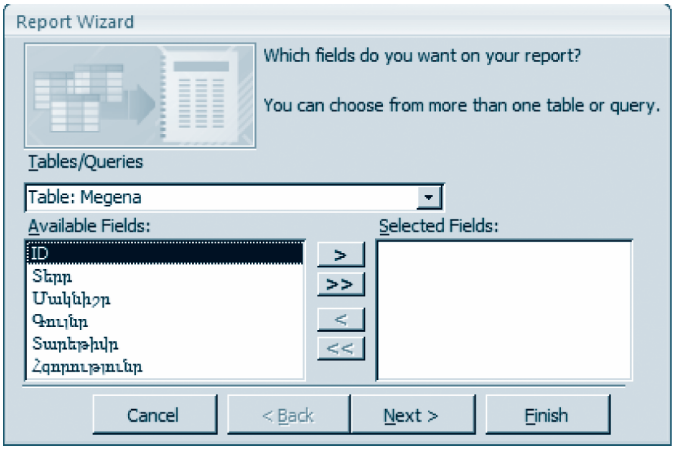
Հաշվետվությունները տվյալների հենքում ներառված ինֆորմացիան անհրաժեշտ տեսքով տպագրելու համար նախատեսված միջոցներ են տրամադրում: Աղյուսակների կամ հարցումների դաշտերի տվյալներն արտածելուց բացի, սրանց միջոցով կարելի է նաև տարբեր արդյունարար արժեքներ հաշվարկել, պետք եղած տվյալները խմբավորել և այլն:

Access-ում *հաշվետվություն ստեղծելու 3* հիմնական եղանակ կա. *ավտոհաշվետվության* միջոցով, *կոնստրուկտորի ռեժիմում* և *վարպետի օգնությամբ*:

Ծանոթանանք *վարպետի օգնությամբ* հաշվետվություն ստեղծելու եղանակին: *Վարպետի օգնությամբ հաշվետվություն ստեղծելու* համար նախ անհրաժեշտ է ընտրել *Create* ներդիրի *Reports* խմբի (նկ. 1.31.) *Report Wizard* կոճակը:



Նկ. 1.31. Create ներդիրի Reports խումբ



Նկ. 1.32. Դաշտերի ընտրության պատուհան

Վարպետի օգնությամբ հաշվետվություններ ստեղծելու հետագա գործընթացը բաղկացած է *6* փուլից:

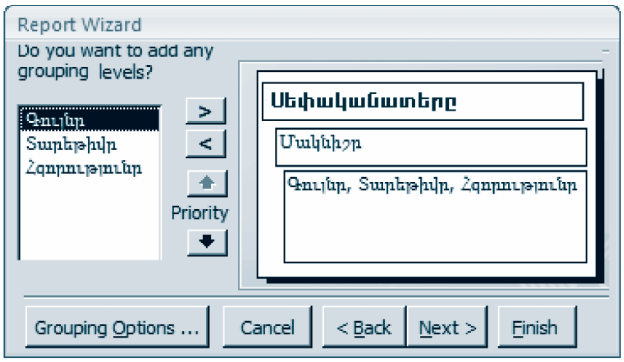
Առաջին փուլում պետք է ընտրել աղյուսակի կամ հարցման այն դաշտերը, որոնք պետք է աղբյուր հանդիսանան ստեղծվող հաշվետվության համար: Դրա համար անհրաժեշտ է.

- մկնիկի ցուցիչը տեղադրել բացված *Report Wizard* պատուհանի (նկ. 1.32.) *Tables/Queries* դաշտի աջ մասի ▼ սլաքի վրա, սեղմել ձախ սեղմակն ու բացված ցուցակից ընտրել անհրաժեշտ աղյուսակը կամ հարցումը,

- *Available Fields* դաշտում նշել ընտրված աղյուսակի կամ հարցման այն դաշտերը, որոնք պետք է ներառել ստեղծվող հաշվետվությունում և սեղմել *Next* կոճակը:

Այստեղ >, >>, < և << կոճակներն ունեն նույն նշանակությունը, ինչ *Form Wizard* պատուհանում:

Երկրորդ փուլում պետք է տալ տվյալները ներկայացնելու տեսքը, այսինքն՝ տվյալները խմբավորել: Հաշվետվությունում կարելի է մինչև 4 մակարդակի խմբեր առանձնացնել: Հերթական մակարդակն ավելացնելու համար անհրաժեշտ է *Report Wizard* պատուհանի (նկ. 1.33.) ձախ մասի ցուցակից նախ ընտրել այն դաշտի անվանումը, ըստ որի ցանկալի է գրառումները խմբավորել, ապա սեղմել > կոճակը: Պատուհանի աջ մասում արտացոլվում է ստեղծվող հաշվետվության կառուցվածքը: Նկ. 1.33.-ում բերված օրինակում որպես խմբավորման վերին մակարդակ ընտրվել է *Սեփականատերը* դաշտը, իսկ որպես երկրորդ մակարդակ՝ *Մակնիշը* դաշտը: Աղյուսակի մնացած դաշտերը արտացոլվել են վերջին, երրորդ մակարդակում: Խմբավորման որևէ մակարդակ կարելի է հեռացնել < կոճակով:



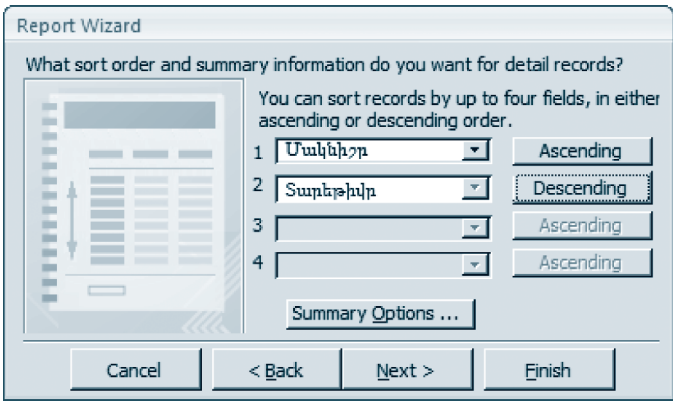
Նկ. 1.33. Տվյալների խմբավորման պատուհան

Նշենք, որ կառուցվածքի մի քանի մակարդակների դեպքում (ինչպես նկ. 1.33.-ում բերված օրինակում) և կոճակներով կարելի է մակարդակները տեղափոխել: Հաջորդ փուլին անցնելու համար պետք է սեղմել *Next* կոճակը:

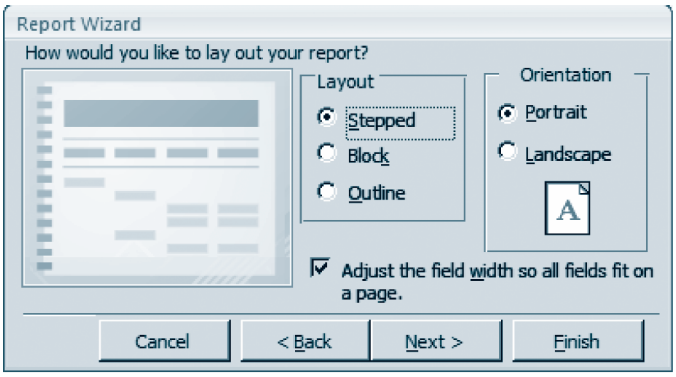
Երրորդ փուլում հաշվետվությունների վարպետը տվյալները կարգավորելու հնարավորություն է տալիս: Ինչպես երևում է նկ. 1.34.-ում, կարելի է մինչև չորս դաշտեր կարգավորել: Տվյալները կարգավորելու համար պետք է համապատասխան դաշտում ընտրել անհրաժեշտ գրառումը, ապա կարգավորման ձևը՝ *Ascending* – աճման կարգով, *Descending* – նվազման կարգով: Չորրորդ փուլին անցնելու համար կրկին պետք է սեղմել *Next*-ը:

Չորրորդ փուլում առաջարկվում է ընտրել հաշվետվության հետևյալ ստանդարտ տեսքերից որևէ մեկը՝ *Stepped*, *Block* կամ *Outline* և որոշել հաշ-

վետվության արտածման ձևը (*Portrait* – երկայնակի, *Landscape* – լայնակի, նկ. 1.35.): Այս ռեժիմում հաշվետվությունը մասշտաբավորվում է այնպես, որպեսզի տեղավորվի էջի լայնքով: *Next*-ի միջոցով անցում կատարեք հինգերորդ փուլ:



Նկ. 1.34. Տվյալների կարգավորման պատուհան

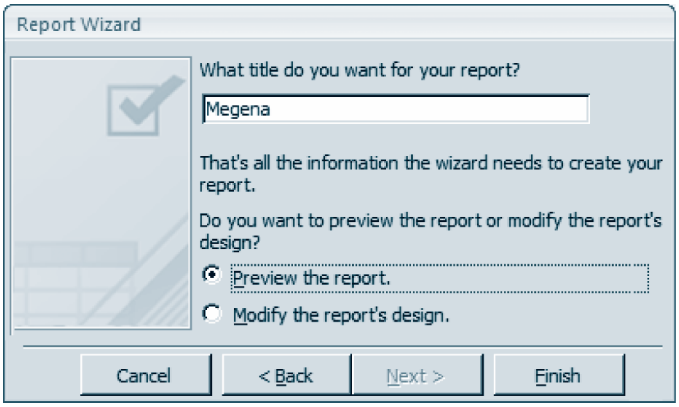


Նկ. 1.35. Հաշվետվության ներկայացման եղանակի ընտրություն

Հինգերորդ փուլում հաշվետվությունների վարպետն առաջարկում է ընտրել հաշվետվության ձևավորման ոճը (նկ. 1.36.-ում ընտրվել է *Office* ոճը): 6-րդ փուլին անցնելու համար կրկին պետք է ընտրել *Next* կոճակը:

Վեցերորդ փուլում առաջարկվում է պահպանել ստեղծված հաշվետվությունը: Դրա համար անհրաժեշտ է *Report Wizard* պատուհանում (նկ. 1.37.) ներմուծել ձևի անվանումը կամ համաձայնել առաջարկվող անվանը: Ստեղծված հաշվետվությունը դիտելու համար նախատեսված է նախնական դիտման ռեժիմը՝ *Preview the report*: Եթե հաշվետվությունը ստեղծելուց հետո ցանկանում եք այն կոնստրուկտորի ռեժիմում փոփոխման ենթարկել, ապա պետք է ընտրել *Modify the report's design* ռեժիմը: Հաշվետվություն ստեղծելու վերջին փուլը պետք է ավարտել *Finish* կոճակով:

Նկ. 1.36. Հաշվետվության ձևավորման ոճի ընտրություն



Նկ. 1.37. Հաշվետվության պահպանման պատուհան

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ *Create* ներդիրի Labels կոճակը հնարավորություն է տալիս տարբեր ստանդարտների փոստային պիտակներ ստեղծել:

1. **Ինչ նպատակով են ստեղծվում հաշվետվությունները:**
2. **Հաշվետվությունների ստեղծման քանի՞ միջոց գիտեք:**
3. **Հաշվետվությունների վարպետը քանի՞ դաշտերում է հնարավորություն տալիս տվյալները կարգավորել:**

Լաբորատոր աշխատանք 1.6
Հաշվետվության ստեղծում

1. Մտեք տվյալների հենքերի ղեկավարման *Access 2007* համակարգի միջավայր:
2. Բացեք նախորդ լաբորատոր աշխատանքում ձեր ստեղծած տվյալների հենքը:
3. Ընտրեք անցումների տիրույթի *Meqena* անվանումով աղյուսակն ու վարպետի օգնությամբ հաշվետվություն ստեղծելու համար սեղմեք *Create* ներդիրի *Reports* խմբի *Report Wizard* կոճակը:
4. Բացված *Report Wizard* պատուհանի *Tables/Queries* դաշտում ընտրեք *Meqena* անվանումով աղյուսակը:
5. *Available Fields* դաշտում հաջորդաբար ընտրեք *Սեփականատերը, Մակնիշը, Տարեթիվը* դաշտերն ու սեղմեք > կոճակը:
6. Հաջորդ փուլ անցնելու համար սեղմեք *Next* կոճակը:
7. *Report Wizard* պատուհանի ձախ մասի ցուցակից ընտրեք *Սեփականատերը* դաշտն ու սեղմեք > կոճակը:
8. Հաջորդ փուլ անցնելու համար սեղմեք *Next* կոճակը:
9. Բացված պատուհանի առաջին դաշտում ընտրեք *Տարեթիվը* դաշտը, ապա կարգավորման *Ascending* տարբերակը:
10. Հաջորդ փուլին անցնելու համար սեղմեք *Next* կոճակը:
11. Ընտրեք հաշվետվության *Stepped* ստանդարտ տեսքը, արտածման *Portrait* ձևն ու *Next* կոճակով հաջորդ փուլ անցեք:
12. Բացված պատուհանում ընտրեք հաշվետվության *Office* ոճն ու *Next* կոճակով հաջորդ փուլ անցեք:
13. Համաձայնեք հաշվետվության համար առաջարկվող անվանն ու հաշվետվության ստեղծումն ավարտեք *Finish* կոճակով:
14. Բացված պատուհանում ներմուծեք ստեղծվող ձևի *Meqena1* անվանումն ու ստեղծվող ձևը փոփոխելու հնարավորություն ունենալու համար ընտրեք *Modify the form's design* փոխանջատիչը:
15. Սեղմեք *Finish* կոճակը, ապա բացված պատուհանում հաջորդաբար ընտրեք ստեղծված դաշտերից յուրաքանչյուրն ու մկնիկի տեղաշարժ-

մամբ փոփոխեք դաշտի չափերն ու դրանց դիրքերն այնպես, որ ստանաք հետևյալ տեսքով հաշվետվություն.

Տերը	Տարեթիվը	Մակնիշը
Անդրեասյան Սոս	2004	BMW528
Անտոնյան Լևոն	1998	Audi A4
Գևորյան Գևորգ	1999	Nissan Altima
Կարապետյան Աշոտ	2003	Volkswagen Passat
Նազարյան Պետրոս	1994	Audi 80
Շահինյան Լևոն	2008	Suzuki Vitara
Պետրոսյան Գոռ	2005	Mersedes_Benz C320
Սիմոնյան Կարեն	2007	Toyota Corolla

16. Ստեղծված տվյալների հենքը պահպանեք *Lab_1_6_** անվանումով, որտեղ *–ի փոխարեն պետք է ներմուծել աշակերտի դասամատյանի համարը:
17. Ավարտեք աշխատանքը հենքերի ղեկավարման *Access* համակարգի հետ՝ օգտվելով համակարգի հիմնական աշխատանքային պատուհանի փակման  սեղմակից:

2. ԾՐԱԳՐԱՎՈՐՄԱՆ *ՊԱՍԿԱԼ* ԼԵԶՎԻ ՀԻՄՈՒՆՔՆԵՐԸ

§2.1. ԾՐԱԳՐԱՎՈՐՄԱՆ ԼԵԶՈՒՆԵՐ

Ինչպես գիտեք, համակարգչում մշակվող ինֆորմացիան ներկայացվում է զրոների և մեկերի միջոցով: Դա է պատճառը, որ հաշվիչ տեխնիկայի զարգացման նախնական փուլում (անցած դարի *40-50*-ական թվականներին), համակարգչին տրվող հրամանների հաջորդականությունն ի սկզբանե ձևակերպվում էր մեքենային հասկանալի զրոների ու մեկերի, այլ կերպ ասած, *մեքենայական կոդերի* միջոցով:

Մեքենայական կոդով տրված հրամանը բաղկացած է այդ հրամանի (գործողության) կոդից և հրամանին մասնակից *օպերանդների* հասցեներից:

Մեքենայական կոդով աշխատելն աշխատատար և ժամանակ պահանջող գործընթաց էր, և մարդ-մեքենա երկխոսությունը մատչելի դարձնելու նպատակով *ծրագրավորման տարբեր լեզուներ* մշակվեցին:

Ծրագրավորման լեզուները **համակարգչի հետ հաղորդակցվելու նպատակով ստեղծված** *արհեստական լեզուներ* **են:**

20-րդ դարի *50*-ական թվականների առաջին կեսին *Ասեմբլեր* (*Assembly language*) անվամբ ծրագրավորման լեզուներ ստեղծվեցին, որոնք զրոներից և մեկերից բացի՝ նաև մարդկային լեզվին մոտիկ հրամաններ, այսինքն՝ *օպերատորներ* էին ներառում: Այս լեզուներում առաջին անգամ սկսեցին *փոփոխականներ*՝ տարբեր արժեքներ կրելու հատկությամբ օժտված մեծություններ կիրառել:

Ասեմբլերով գրված ծրագիրը մեքենայի վրա իրագործելու համար պետք է թարգմանվեր, ուստի այդ նպատակով առաջին անգամ հատուկ թարգմանիչ ծրագիր ստեղծվեց՝ Ասեմբլեր լեզվի *կոմպիլյատորը*:

Կոմպիլյատորները **ծրագրավորման որևէ լեզվով գրված ծրագրերը թարգմանելու նպատակով ստեղծված միջոցներ են, որոնք ստուգում են գրված ծրագրի քերականությունն ու շարահյուսությունը, և սխալ չհայտնաբերելու դեպքում այն վերածում մեքենայական կոդի:**

Ասեմբլերային լեզուները ստեղծվում էին կոնկրետ տիպի պրոցեսորների համար և հարմարեցվում էին դրանց առանձնահատկություններին: Այդ պատճառով նման լեզուներն անվանեցին ծրագրավորման *ցածր մակարդակի* լեզուներ՝ նկատի առնելով, որ նման լեզվի հրամանները շատ մոտ են մեքենայական կոդին:

50-ական թթ. երկրորդ կեսից սկսեցին ստեղծվել ծրագրավորման *բարձր մակարդակի* լեզուներ: Սրանք բնական լեզվին զգալիորեն մոտ լեզուներ են և առավել հասկանալի մարդուն, քան համակարգչին: Այս լեզուներն արդեն, ի տարբերություն ցածր մակարդակի լեզուների, կախված չեն համակարգչի տիպից:

Տարբեր խնդիրներ լուծելու նպատակով հետագայում բարձր մակարդակի ծրագրավորման զանազան լեզուներ ստեղծվեցին՝ *FORTRAN, COBOL, BASIC* և այլն: Ծրագրավորման *Fortran* (*FORmula TRANslator*) լեզուն նախատեսված էր գիտատեխնիկական հաշվարկներ կատարելու համար: *COBOL* (*COmmon Business – Oriented Language*) – նախատեսված էր կոմերցիային առնչվող խնդիրներ լուծելու համար: *BASIC* (*Beginner's All – Purpose Symbolic Instruction Code*) լեզուն աչքի էր ընկնում ուսուցման պարզությամբ և ուղղված էր սկսնակ ծրագրավորողներին:

80-ական թթ. սկզբին ծրագրավորման մեջ նոր, այսպես կոչված, *ալգորիթմական լեզուներ* մշակվեցին, որոնք սկիզբ հանդիսացան *կառուցվածքային ծրագրավորման* համար: Այս լեզուների հիմնական առանձնահատկությունն այն է, որ ալգորիթմական կառույցներն արդյունավետորեն ծրագրավորելու հնարավորություն ընձեռեցին:

Կառուցվածքային ծրագրավորման կայացման գործում մեծ դեր ունի ինչպես *Pascal* (*Պասկալ*) լեզուն, որը մեր դասընթացի մասն է կազմելու, այնպես էլ լայնորեն հայտնի *C(Սի)* լեզուն, որը թույլատրում է արագագործ ծրագրեր ստեղծել: *Պասկալ* ալգորիթմական լեզվի հիման վրա ստեղծվեցին *Object Pascal*, իսկ *QBasic* լեզվի հիման վրա՝ *Visual Basic* օբյեկտային կողմնորոշմամբ լեզուները:

C լեզվի հիման վրա *1980* թ. ստեղծվեց *C++ օբյեկտային կողմնորոշմամբ* ծրագրավորման լեզուն: Օբյեկտային կողմնորոշմամբ ծրագրավորման լեզուների առանձնահատկությունն այն է, որ հիմնված են *փվյալները* և դրանք մշակող *մեթոդները* միավորող *ծրագրային օբյեկտների* վրա:

20-րդ դարի *90*-ական թթ. Համացանցի զարգացմանը զուգընթաց ծրագրավորման այնպիսի նոր լեզուներ ստեղծվեցին, որոնք թույլատրում են օպերացիոն տարբեր հենքերի վրա աշխատող ծրագրեր ստեղծել: Այսինքն՝ միևնույն ծրագիրը կարող է աշխատել Համացանցին միացված ցանկացած համակարգչի վրա՝ անկախ դրա վրա եղած ընթացիկ օպերացիոն համակարգից (*WINDOWS, LINUX, Mac OS* և այլն): Այդպիսի լեզու է օբյեկտային կողմնորոշմամբ *Java* (*Ջավա*) լեզուն, որը ստեղծվել է *C++* -ի հիմքում:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *Java* լեզուն ներկայումս աշխարհում լայնորեն կիրառվող երկրորդ լեզուն է՝ *Basic*-ից հետո:

1. **Ինչպե՞ս էին կազմվում առաջին համակարգչային ծրագրերը:**
2. **Ի՞նչ են ներկայացնում ծրագրավորման լեզուները:**
3. **Ի՞նչ է կոմպիլյատորը. առաջինը ո՞ր լեզվի համար այն ստեղծվեց:**
4. **Առաջին ծրագրավորման լեզուներն ինչո՞ւ կոչվեցին ցածր մակարդակի:**
5. **Ծրագրավորման բարձր մակարդակի մի քանի լեզուներ թվարկեք:**
6. **Ձեր կարծիքով, *Պասկալ* լեզուն ի՞նչ տիպի ծրագրավորման լեզու է:**
7. **Ո՞րն է օբյեկտային կողմնորոշմամբ լեզուների գլխավոր առանձնահատկությունը:**

§2.2. ԾՐԱԳՐԱՎՈՐՄԱՆ *ՊԱՍԿԱԼ* ԼԵԶՈՒ: ԼԵԶՎԻ ԱՇԽԱՏԱՆՔԱՅԻՆ ՄԻՋԱՎԱՅՐԸ

Ծրագրավորման *Պասկալ* լեզուն ստեղծվել է 1968-71 թթ.՝ շվեյցարացի գիտնական *Նիկլաուս Վիրթի* կողմից, և անվանվել ֆրանսիացի նշանավոր մաթեմատիկոս, փիլիսոփա *Բլեզ Պասկալի* անունով: Ի սկզբանե լեզուն ստեղծվել է որպես ուսուցողական՝ նպատակ ունենալով հնարավորինս մատչելի դարձնել ծրագրավորման հիմունքների ուսուցումը: Ստեղծված լեզուն իր պարզ քերականության և կիրառման ոլորտների բազմազանության շնորհիվ շուտով դարձավ ժամանակի ամենատարածված լեզուներից մեկը:

Ներկայացնենք *Պասկալ* լեզվի հիմնական առանձնահատկություններից մի քանիսը.

- *Լեզվի պարզ քերականությունը:* Հիմնական հասկացությունների փոքրաթիվությունը: *Պասկալ*-ով գրված ծրագրի ընթեռնելիության պարզությունը:
- *Պասկալ*-ի կոմպիլյատորի և այդ լեզվով գրված ծրագրի կողմից հաշվիչ համակարգին ներկայացվող բավական *պարզ պահանջները:*

- *Լեզվի ունիվերսալությունը.* *Պասկալ* լեզուն գործնականում կիրառելի է ծրագրավորման ցանկացած տիպի խնդիրների համար (հաշվարկային, տնտեսագիտական և այլն):

Մենք ուսումնասիրելու ենք լեզվի *Տուրբո Պասկալ* (հետագայում՝ *Պասկալ*) տարբերակը, որը մշակվել է *1992* թվականին՝ *Borland International* ֆիրմայի կողմից: Այս տարբերակի կոմպիլյատորը (ինչպես բոլոր *Turbo* կոմպիլյատորները) բավական արագագործ է:

Լեզվի աշխատանքային միջավայրը թույլատրում է.

- ստեղծել ծրագրի տեքստը,
- իրագործել ծրագրի կոմպիլյացիան,
- օպերատիվ կերպով ուղղել գտնված քերականական սխալները,
- կարգաբերել ծրագիրն ու իրականացնել այն,
- օգտվել օգնության ներկառուցված համակարգից:

Ծրագրի աշխատանքային միջավայր մտնելու համար անհրաժեշտ է.

- համակարգչի կոշտ սկավառակների վրա գտնել *TP, TURBO, TURBOPAS* կամ *PASCAL* անվանումներից որևէ մեկը կրող թղթապանակն ու բացել այն,
- բացված թղթապանակում ընտրել *Turbo.exe* ֆայլն ու կատարման տալ (սեղմել *Enter* ստեղնը կամ մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարել):

Այս գործողությունների արդյունքում էկրանին բերվում է լեզվի աշխատանքային միջավայրը (նկ. 2.1.):

Ծանոթանանք բացված պատուհանի հետևյալ երեք տեղամասերին.

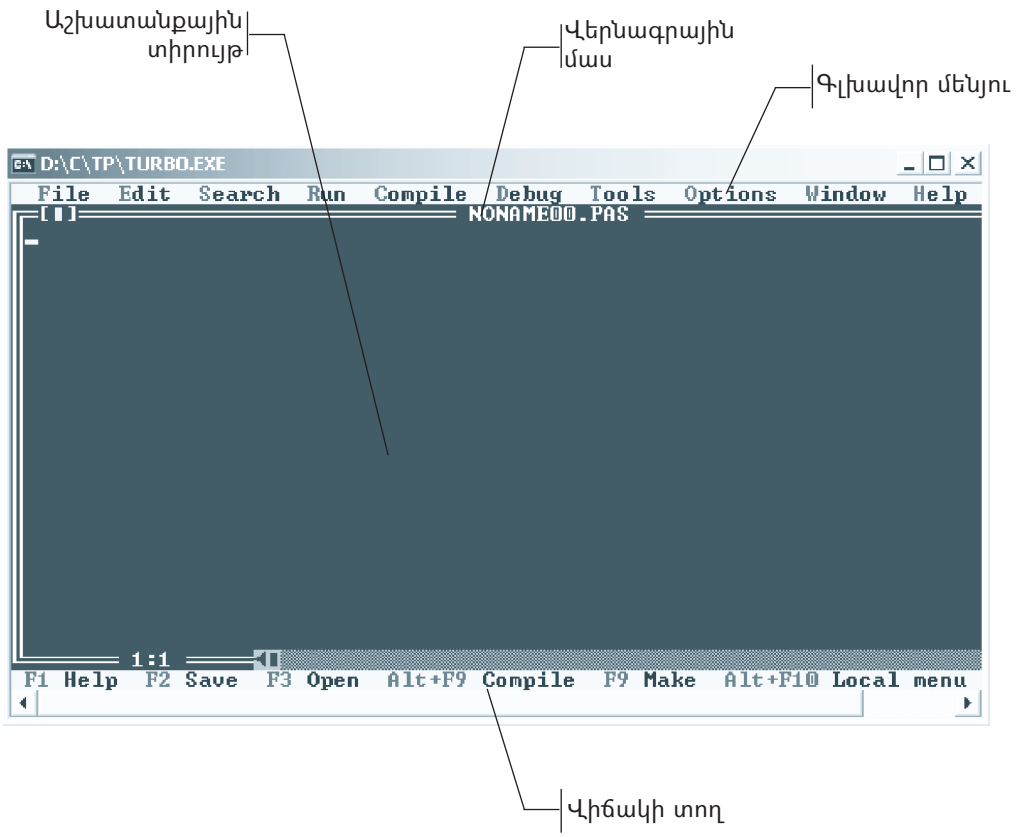
- գլխավոր մենյու,
- աշխատանքային տիրույթ,
- վիճակի տող:

Գլխավոր մենյու կարելի է մտնել երկու եղանակով՝ սեղմելով ստեղնաշարի վերին մասում առկա *F10 ֆունկցիոնալ ստեղնը*, կամ օգտվելով մկնիկի ձախ սեղմակից:

Գլխավոր մենյուն պարունակում է հետևյալ բաղկացուցիչ միջոցները.

- *File* – ֆայլի հետ կապված հնարավոր գործողություններ է ընդգրկում (նոր ֆայլ ստեղծել (*New*), նախկինում ստեղծվածը բացել (*Open*), աշխատանքային միջավայրում առկա ծրագիրը պահպանել (*Save, Save as...*) և այլն:
- *Edit* – օգնում է ծրագրի տեքստը խմբագրել *Copy* - պատճենել, *Paste* – տեղադրել, *Delete* – ընտրված հատվածը ջնջել, *Undo/Redo* – խմբագրական վերջին գործողությունը *անփեսել/վերականգնել* և այլն:
- *Search* – ծրագրի տեքստում թույլատրում է որոնել ներմուծված նմուշի կրկնօրինակները (*Search*) և անհրաժեշտության դեպքում դրանք փոխարինել (*Replase*) նոր տեքստով:

- *Run* – թույլատրում է հնարավոր մի քանի ռեժիմներով գրված ծրագիրն իրագործել:
- *Compile* – ծրագիրը կոմպիլյացիայի ենթարկելու մի քանի եղանակներ է պարունակում:
- *Debug* – հնարավորություն է տալիս ծրագրում առկա տրամաբանական սխալները գտնել:
- *Tools* – որոշ լրացուցիչ միջոցներ է պարունակում:
- *Options* – թույլատրում է սահմանել կոմպիլյացիայի և ինտեգրացված միջավայրի նախագծման համար անհրաժեշտ պարամետրերը:
- *Window* – թույլատրում է իրականացնել պատուհանների հետ կատարվող բոլոր հիմնական գործողությունները (բացել, փակել, տեղաշարժել, չափերը փոփոխել):
- *Help* – թույլատրում է օգտվել համակարգչի օգնության ներքին համակարգից:



Նկ. 2.1. Տուրբո Պասկալի աշխատանքային միջավայրը

Գլխավոր մենյուի ցանկացած բաղադրիչ կարելի է ակտիվացնել նաև *Alt* ստեղնի և այդ բաղադրիչի մեջ առկա կարմիր գույնի պայմանանիշի միաժամանակյա սեղմմամբ (օրինակ՝ *ALT-F*-ը բացում է *File* մենյուն):

Պատուհանի *աշխատանքային տիրույթում* կարելի է ներմուծել ծրագրի տեքստը, այն խմբագրել և այլն: Պատուհանի *վերնագրային* մասում գրվում է ծրագրի տեքստը պարունակող ֆայլի անվանումը (ծրագրի նոր ստեղծվող տեքստին ավտոմատ տրվում է *NONAME* անվանումը՝ *NONAME00.PAS, NONAME01.PAS* և այլն):

Վիճակի տողը որոշ կարևոր գործողություններ և դրանց համապատասխանող ստեղների համադրություններ է պարունակում:

Ծանոթանանք *Պասկալի* միջավայրում աշխատելու հիմնական սկզբունքներին: Ծրագրի տեքստը ներմուծվում է ստեղնաշարից, ընդ որում՝ խմբագրիչի պատուհանին անընդհատ թարթող *կուրսորը* ցույց է տալիս ներմուծվող պայմանանշանի դիրքը: Յուրաքանչյուր տող ներմուծելուց հետո հաջորդ տողին անցում է կատարվում *ENTER*-ի միջոցով: Տողն ամենաշատը կարող է *126* պայմանանշան պարունակել:

Պասկալի միջավայրում խմբագրիչի պատուհանը կարող է ըստ ընտրված ռեժիմի պարունակել *25* կամ *50* տող:

Ներմուծված տեքստի վրայով հնարավոր է տեղաշարժվել հետևյալ ստեղների միջոցով.

Home – վերադարձ ընթացիկ տողի սկզբնամաս,
End – անցում ընթացիկ տողի վերջնամաս,
PgUp – ընթացիկ տողից մեկ էջի չափով (*25* կամ *50* տող) տեղաշարժվել ծրագրի սկզբնամաս (վերև),
PgDn – ընթացիկ տողից մեկ էջի չափով տեղաշարժվել ներքև,
Ctr-PgUp – վերադարձ ծրագրի սկիզբ,
Ctr-PgDn – անցում ծրագրի վերջին:

Ստեղնաշարի ←, ↑, →, ↓ ստեղների միջոցով կուրսորը կարելի է մեկ պայմանանշանի չափով տեղաշարժել համապատասխանաբար դեպի ձախ, վերև, աջ կամ ներքև:

Կուրսորի ընթացիկ դիրքից անմիջապես ձախ ընկած պայմանանշանը կարելի է ջնջել *Backspace*, իսկ ընթացիկ պայմանանշանը՝ *Delete* ստեղնի օգնությամբ:

Սխալմամբ ջնջված ինֆորմացիան կարելի է քայլ առ քայլ կրկին վերականգնել *Alt-Backspace* ստեղների համատեղ հաջորդական սեղմումներով:

Երբ ծրագրի տեքստը ներմուծված է, այն պետք է ենթարկել *թարգմանման* և *իրագործման*. դրա համար կարելի է օգտվել *Ctr-F9* ստեղներից:

Ծրագրի աշխատանքն ավարտելուց հետո էկրանին կրկին բերվում է խմբագրիչի պատուհանը (աշխատանքային միջավայրը):

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *Տուրբո Պասկալի* ինտեգրացված միջավայրում աշխատելիս կարելի է օգտվել հետևյալ հիմնական հրամաններից և դրանց հետ կապված արագագործ ստեղներից.

  Alt+F5 (*Alt* և *F5* ստեղների համատեղ սեղմում) – ծրագրի աշխատանքի արդյունքի դիտում,

  F2 – աշխատանքային տիրույթում առկա ծրագրի տեքստի պահպանում,

  Alt + X – ելք Տուրբո Պասկալի միջավայրից,

  F1 – օգնության ներքին համակարգի կանչ,

  Ctrl + F1 – կուրսորի միջոցով ընտրված հրամանի վերաբերյալ տեղեկույթի տրամադրում,

  Ctrl + Y – ընթացիկ տողի ջնջում:

1. **Ինչ նպատակով է նախագծվել *Պասկալ* լեզուն:**
2. **Թվարկեք *Պասկալ* լեզվի ձեզ հայտնի առանձնահատկությունները:**
3. ***Պասկալ*-ի աշխատանքային միջավայրն ինչ հնարավորություններ է ընձեռում:**
4. **Թվարկեք *Պասկալի* միջավայրի գլխավոր մենյուի ձեզ հայտնի բաղկացուցիչ մասերը:**
5. **Ինչի՞ համար է միջավայրի խմբագրիչը:**
6. **Ծրագրի տեքստով տեղաշարժվելու ինչ միջոցներ գիտեք:**
7. **Ինչպե՞ս կարելի է վերականգնել սխալմամբ ջնջված վերջին ինֆորմացիան:**

§2.3. ՊԱՍԿԱԼ ԼԵԶՎԻ ՏԱՐՐԵՐԸ

Պասկալ լեզվի այբուբենը բաղկացած է *տառերից*, *թվանշաններից*, *հատուկ պայմանանշաններից* և *առանցքային բառերից*:

Լեզվում կիրառում են *լատինական այբուբենի մեծատառերն* ու *փոքրատառերը.*

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z

Պասկալ լեզուն տարբերություն չի դնում մեծատառերի և փոքրատառերի միջև, օրինակ՝ *VAR* և *var* բառերը միևնույն նշանակությունն ունեն:

Լեզվում օգտագործվում են արաբական *0, 1, 2, ..., 9* թվանշանները:

Հատուկ պայմանանշանները, կախված կիրառումից, կարող են տարբեր իմաստներ ունենալ.

{ } [ ] ( ) ‘ ; , . = + - * / : _ = > < # & @

Լեզվում օգտագործում են նաև *պայմանանշանների* որոշակի *համադրություններ*, որոնք ընդունվում են որպես ամբողջություն, գրվում են կից՝ առանց բացատանիշի և որոշակի իմաստ ունեն.

(* *) := >= <= <> . . .

Պասկալում ընդունված *առանցքային բառերը* որոշակի իմաստ ունեցող հատուկ տերմիններ են:

Լեզվում կիրառվում են հետևյալ առանցքային բառերը.

and	*downto*	*inline*	*procedure*	*type*
array	*else*	*interface*	*program*	*unit*
asm	*end*	*label*	*record*	*until*
begin	*file*	*mod*	*repeat*	*uses*
case	*for*	*nil*	*set*	*var*
const	*function*	*not*	*shl*	*while*
constructor	*goto*	*object*	*shr*	*with*
destructor	*if*	*of*	*string*	*xor*
div	*implementation*	*or*	*then*	
do	*in*	*packed*	*to*	

Ծրագիր կազմելիս հաճախ է անհրաժեշտ լինում որոշ մեծություններին անվանումներ տալ և հետագայում դրանք այդ անվանումներով կիրառել: Տրված անվանումները կոչվում են *իդենտիֆիկատորներ*: Իդենտիֆիկատորները կազմվում են լեզվում ընդունված տառերով ու թվերով, իսկ հատուկ պայմանանշաններից կարող են պարունակել միայն ընդգծման (_) նշանը:

Իդենտիֆիկատորը *պետք է սկսվի տառով* **և** *չի կարող առանցքային բառ լինել*:

Հետևյալ գրառումները ճիշտ իդենտիֆիկատորներ են՝ *A2B, DDd, A_3*:

Քանի որ լեզուն մեծատառերի և փոքրատառերի միջև տարբերություն չի դնում, ապա *BETA, Beta* և *beta* իդենտիֆիկատորները համարժեք են և նշում են միևնույն մեծությունը:

Եթե իդենտիֆիկատորը պետք է մի քանի բառ պարունակի, ապա կարելի է բաղադրիչ բառերը սկսել մեծատառերով կամ դրանց միջև ընդգծման նշան դնել: Օրինակ՝ *AlfaBeta կամ ALFA_BETA:*

Այն մեծությունները, որոնց արժեքները ծրագրի կատարման ընթացքում չեն փոփոխվում, կոչվում են *հաստատուն մեծություններ* **կամ** *հաստատուններ*:

Հաստատունները ծրագրում կարող են ներկայացվել ինչպես կոնկրետ արժեքների, այնպես էլ անվանումների (իդենտիֆիկատորների) միջոցով:

Ծանոթանանք *Պասկալում* կիրառվող *թվային* և *սիմվոլային* տիպերի հաստատուններին:

Թվային հաստատունները գրվում են մաթեմատիկայից ձեզ հայտնի գրելաձևով, ընդ որում՝ դրական թվի դեպքում + նշանը կարող է բացակայել:

Թվերը կարող են ներկայացվել *ամբողջ* և *իրական* թվերի տեսքով: Իրական թվում ամբողջ և կոտորակային մասերն իրարից անջատվում են *փասնորդական կետի* (.) միջոցով: Իրական թիվը *չի կարող* սկսվել կամ ավարտվել կետով: *Պասկալի* կանոններով՝ ճիշտ հաստատուն թվային մեծություններ են, օրինակ՝ *1900, +5, -7, 2.85, 0.02, -10.802, -6.0* թվերը:

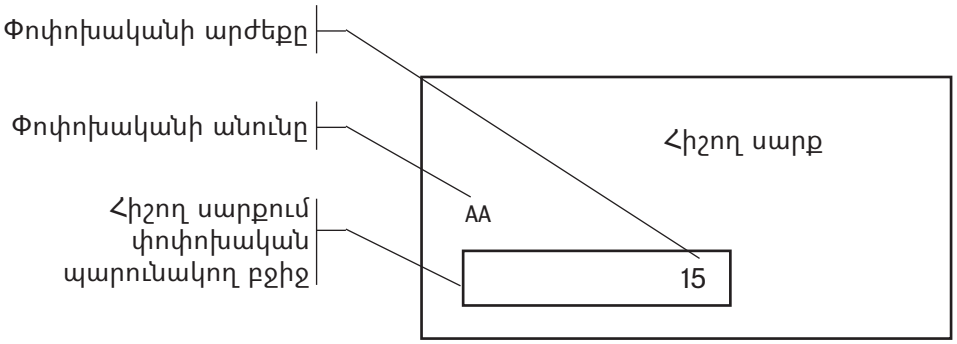
Իրական թվերը ներկայացվում են նաև մեկ այլ՝ *ցուցչային եղանակով*, որտեղ թվի *10*-ական կարգը գրվում է *E* կամ *e* տառից հետո: Օրինակ՝ *0.003* կամ $0.3 \cdot 10^{-2}$ թիվը այս եղանակով արտահայտելիս կարելի է գրել *0.3E-2* կամ *0.03E-1* տեսքով:

Սիմվոլային հաստատունը ստեղնաշարին առկա ցանկացած *տառ*, *պայմանանշան* կամ *թվանշան* է՝ առնված ապաթարցերի մեջ: Օրինակ՝ *'c'* – *c* տառն է, իսկ *')'*–ը փակող փակագիծը:

Ցանկացած ծրագիր գրելիս անհրաժեշտ է լինում մուտքային և որոշ ընթացիկ տվյալներ հետագա մշակման նպատակով պահպանել: Դրա համար օգտվում են *փոփոխականներից*:

Այն մեծությունները, որոնց արժեքները ծրագրի կատարման ընթացքում կարող են փոփոխվել, կոչվում են *փոփոխականներ*:

Փոփոխականները ծրագրում ներկայացվում են *անունների* (իդենտիֆիկատորների) միջոցով: Յուրաքանչյուր *փոփոխականի անունը եզակի է* և ծրագրի կատարման ընթացքում *չի կարող փոփոխվել*: Ընդ որում՝ համակարգչի մեջ յուրաքանչյուր փոփոխականի համար համապատասխան ծավալով հիշողություն է հատկացվում (նկ. 2.2.):



Նկ.2.2. Փոփոխականի տեղակայումը համակարգչի հիշող սարքում

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Իդենտիֆիկատորը կարող է ցանկացած երկարություն ունենալ, սակայն *Պասկալ* լեզվի համար նշանակալից են միայն դրա առաջին *63* պայմանանշանները:

1. **Ո՞ր լեզվի այբուբենն է կիրառվում *Պասկալ* լեզվում:**
2. **Ինչ է իդենտիֆիկատորը:**
3. **Նշեք, թե հետևյալ իդենտիֆիկատորներից որո՞նք են սխալ և ինչո՞ւ.**
   ա) *X – 4*, **բ)** *eps7*, **գ)** *Alfa 1*, **դ)** *ParsKar*,
   ե) *Pars_Kar*, **զ)** *3h26*, **է)** *a&b:*
4. **Ո՞ր մեծություններն են անվանում հաստատուն:**
5. **Ինչ է սիմվոլային հաստատունը:**
6. **Ո՞ր մեծություններն են կոչվում փոփոխական մեծություններ, և ինչպե՞ս են դրանք ծրագրում ներկայացվում:**

§2.4. ՊԱՍԿԱԼ ԾՐԱԳՐԻ ԿԱՌՈՒՑՎԱԾՔՆ ՈՒ ՀԻՄՆԱԿԱՆ ԲԱԺԻՆՆԵՐԸ

Պասկալով գրված ծրագրի հիմնական կառուցվածքը կարելի է ընդհանուր կերպով ներկայացնել հետևյալ կերպ.

ծրագրի վերնագիր,
ծրագրում կիրառվող մեծությունների հայտարարում,
ծրագրի հիմնական մարմին:

Ծրագրի վերնագիրը սկսվում է *PROGRAM* առանցքային բառով, որին հաջորդող մեկ կամ մի քանի բացատանիշից հետո գրվում է *ծրագրի անունը*:

Ծրագրի անունը **ցանկացած իդենտիֆիկատոր է, որն ավարտվում է կետ-ստորակետով (;):**

Օրինակ՝ *PROGRAM KHNDIR;*

Ընդհանրապես ծրագրի վերնագիրը կարելի է չգրել, բաց թողնել:

Ինչպես վերնագիրը, այնպես էլ *Պասկալի* ցանկացած հրաման (օպերատոր) ավարտվում է կետ-ստորակետով (;):

Պասկալով գրված ծրագրում կիրառվող բոլոր մեծությունները նախօրոք պետք է *նկարագրվեն* կամ, որ նույնն է, *հայտարարվեն*: Նկարագրությունների կամ հայտարարությունների մասը կարող է ներառել հետևյալ հնարավոր բաղադրիչները.

նոր տիպերի նկարագրություններ,
հաստատունների նկարագրություններ,
նշիչների նկարագրություններ,
փոփոխականների նկարագրություններ:

Նոր տիպերի նկարագրությունները սկսվում են *TYPE* առանցքային բառով, որին հաջորդում են ստեղծվող *նոր տիպերի* հայտարարությունները՝ իրարից ;-երով փոխանջատված: Նոր տիպերը ստեղծվում են *Պասկալում* սահմանված տիպերի հիման վրա: Օրինակ՝

```
TYPE NOR_TIP=REAL;
     TARIQ=BYTE;
```

որտեղ *NOR_TIP* և *TARIQ* իդենտիֆիկատորները ստեղծվող նոր տիպերի անուններն են, իսկ հավասարման (=) նշանին հաջորդում են *Պասկալում* նախասահմանված *REAL* և *BYTE* տիպերը (սրանց կծանոթանանք հաջորդ պարագրաֆում):

Հաստատունների հայտարարությունները սկսվում են *CONST* առանցքային բառով: Օրինակ՝ *CONST e=2.7; tar='a';* և այլն:

Ըստ այս հայտարարման՝ *e* և *tar* իդենտիֆիկատորները ծրագրի կատարման ընթացքում արժեքները փոփոխել չեն կարող՝ *e*-ի արժեքը կմնա *2.7*, իսկ *tar*-ի արժեքը՝ *'a'* պայմանանշանը:

Նշիչների հայտարարությունը տրվում է առանցքային *LABEL* բառով: Օրինակ՝

```
LABEL cikl, ab, 57;
```

Այստեղ *LABEL* բառից հետո, իրարից ստորակետերով անջատված, թվարկվել են *cikl, ab* և *57* նշիչները:

Նշիչ **կարող է լինել ինչպես իդենտիֆիկատորը, այնպես էլ դրական ամբողջ թիվը:**

Նշիչներ ստեղծվում են, երբ պետք է անուններ տալ այն օպերատորներին (հրամաններին), որոնց պետք է անցում կատարել՝ խախտելով ծրագրի կատարման հաջորդական ընթացքը: Միևնույն նշիչը չի կարող մեկից ավելի օպերատորներ նշել, և հայտարարված նշիչը պետք է ծրագրում օգտագործել:

Օպերատորը նշելու համար անհրաժեշտ է նշիչի և համապատասխան օպերատորի միջև վերջակետ (:) դնել:

Ծրագրում օգտագործված *փոփոխականները* պետք է նախօրոք *հայտարարել VAR* առանցքային բառի տակ: Օրինակ՝

```
VAR c : CHAR;
    x , k : BYTE;
    d , m , l : REAL;
```

Ինչպես երևում է օրինակից, նույն տիպի փոփոխականներ հայտարարելիս կարելի է դրանք խմբավորել՝ իրարից ստորակետերով անջատելով: Իդենտիֆիկատորն ու տիպ արտահայտող առանցքային բառը (*BYTE, REAL* և այլն) իրարից պետք է փոխանջատել վերջակետով (:):

Իրագործվող հրամանների հաջորդականությունը վերցվում է *BEGIN* և *END* առանցքային բառերի միջև և կազմում է *ծրագրի հիմնական մարմինը*:

BEGIN **և** *END* **բառերի միջև վերցված օպերատորների ամբողջությունն անվանում են** *բլոկ*:

Ծրագրում բազմաթիվ բլոկներ կարող են ներառվել, սակայն միայն ծրագրի հիմնական մարմինն ավարտող *END*-ն է ավարտվում կետով (.):

Բլոկները կարող են նաև լինել ներդրված, այսինքն՝ մեկը մյուսի մեջ ներառված:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Թվով արտահայտված նշիչը կարող է [0;9999] միջակայքի որևէ թիվ լինել:
- Միևնույն օպերատորը կարող է մի քանի նշիչներ ունենալ. այս դեպքում դրանք իրարից պետք է անջատել վերջակետերով (:):

1. **Թվարկեք ծրագրի կառուցվածքի բաղադրիչները:**
2. **Որո՞նք են *Պասկալով* գրված ճիշտ վերնագրերը.**
   ա) *Program L_9;* **բ)** *PROGRAM kk+5;*
   գ) *ProGram DD;* **դ)** *Program 5AB;*
3. **Ո՞ր նշիչներն են ճիշտ հայտարարված.**
   ա) *LABEL a;* **բ)** *label -5;* **գ)** *LABLE a6;*
   դ) *LABEL 1.25;* **ե)** *label LL,67,k_8;*
4. **Ո՞ր փոփոխականներն են ճիշտ հայտարարված.**
   ա) *VAR a:b:REAL;* **բ)** *VAR c:CHAR, d:REAL;* **գ)** *VAR 2:BYTE;*
5. **Ինչ է բլոկը: Ո՞ր բլոկն է ավարտվում կետով (.):**

§2.5. ՏՎՅԱԼՆԵՐԻ ՊԱՐԶԱԳՈՒՅՆ ՏԻՊԵՐ: ՄԱԹԵՄԱՏԻԿԱԿԱՆ ՖՈՒՆԿՑԻԱՆԵՐ ԵՎ ԱՐՏԱՀԱՅՏՈՒԹՅՈՒՆՆԵՐ

Համակարգչում ներկայացվող ցանկացած *տվյալ* բնութագրվում է իր *տիպով:*

Պասկալ լեզվում տվյալների *պարզ տիպեր* են համարվում *կարգային* և *իրական* տիպերը: Կարգային տիպի փոփոխականները կարող են վերջավոր քանակությամբ արժեքներ ունենալ, որոնք հնարավոր է համարակալել:

Կարգային տիպերից ուսումնասիրելու ենք *ամբողջ*, *տրամաբանական*, *սիմվոլային* ու *միջակայքային* տիպերը:

Ամբողջ տիպի մեծությունների հնարավոր արժեքները կախված են այդ մեծություններին տրամադրվող հիշողության ծավալից (աղյուսակ 2.1):

Տրամաբանական տիպի փոփոխականի հնարավոր արժեքները երկուսն են՝ *TRUE* (*ճիշտ*) և *FALSE* (*սխալ*): Ընդ որում՝ *FALSE*-ին համապատասխանում է *0* կարգահամարը (արժեքը), իսկ *TRUE*-ին՝ *1*: Տրամաբանա-

կան տիպի մեծությանը տրամադրվում է *1* բայթ: Տրամաբանական տիպի փոփոխականը հայտարարվում է *BOOLEAN* առանցքային բառով, օրինակ՝ *T:BOOLEAN . . . T:=true;:*

Աղյուսակ 2.1

Ամբողջ տիպ		
Տիպի անվանումը	**Տրամադրվող հիշողության ծավալը (բայթերով)**	**Հնարավոր արժեքների միջակայքը**
BYTE	*1*	*0 ÷ 255*
WORD	*2*	*0 ÷ 65535*
INTEGER	*2*	*–32 768 ÷ 32 767*

Սիմվոլային տիպի մեծություն է համակարգչում ներկայացվող ցանկացած պայմանանշան՝ տառ, ապաթարցերի միջև առնված միանիշ թվանշան և ցանկացած հատուկ պայմանանշան: Սիմվոլային տիպի փոփոխականը հայտարարվում է *CHAR* առանցքային բառով:

Օրինակ՝ *c:CHAR; ... c:='a'; c:='7';*

Միջակայքային տիպը հիմնվում է ցանկացած կարգային տիպի վրա և ստացվում է հիմնային հանդիսացող տվյալ կարգային տիպի մի մասից: Այս տիպը տրվում է իր ստորին ու վերին եզրային արժեքների միջոցով՝ դրանք իրարից բաժանելով երկու կետով (..): Օրինակ՝

```
Type t='a'..'d';
       b=1..10;
Var  alpha :t ;
       number :b;
```

Ըստ այս հայտարարության՝ *alpha*-ն կարող է ընդունել *'a', 'b', 'c', 'd'* արժեքները, իսկ *number*-ը՝ *1*-ից *10* թվերից ցանկացածը:

Պասկալը իրական թիվ ներկայացնելու մի քանի *տիպեր* ունի. մենք կկիրառենք միայն *REAL* տիպը (աղյուսակ 2.2):

Աղյուսակ 2.2

Իրական տիպ			
Տիպի անվանումը	**Բայթերի քանակը**	**Թվի բաղադրիչ թվանշանների քանակը**	**Արժեքների միջակայքը**
REAL	*6*	*11–12*	$2.9 \cdot 10^{-39} \div 1.7 \cdot 10^{38}$
DOUBLE	*8*	*15–16*	$5.0 \cdot 10^{-324} \div 1.7 \cdot 10^{308}$
COMP	*8*	*19–20*	$-2 \cdot 10^{63}+1 \div 2 \cdot 10^{63}-1$

Լեզվում կարգային և իրական տիպերի հետ աշխատող մի շարք *ստանդարտ ֆունկցիաներ* կան:

x և *y* ամբողջ տիպի արգումենտների համար սահմանված են հետևյալ ֆունկցիաները.

DEC(x,y) – *x*-ի արժեքը նվազեցնում է *y*-ի չափով, իսկ եթե *y*-ը բացակայում է *(DEC(x))*, ապա *x*-ի արժեքը նվազում է *1*-ով,

INC(x,y) – *x*-ի արժեքը աճում է *y*-ի չափով, իսկ եթե *y*-ը բացակայում է *(INC(x))*, ապա *x*-ի արժեքը աճում է *1*-ով,

ODD(x) – վերադարձնում է *TRUE*, եթե *x*-ը կենտ թիվ է, և *FALSE* – հակառակ դեպքում:

Պասկալում իրական տիպի արգումենտների համար նույնպես որոշակի *մաթեմատիկական ստանդարտ ֆունկցիաներ* կան մշակված: Աղյուսակ 2.3-ում բերվել են դրանց համառոտ նկարագրությունները:

Աղյուսակ 2.3

Մաթ. ֆունկցիան	**Համարժեք տեսքը *Պասկալում***	**Արգումենտի տիպը**	**Արդյունքի տիպը**	**Գործողությունը**
sin x	*SIN(x)*	ամբողջ, իրական	իրական	ռադիանով արտահայտված *x* արգումենտի սինուսը
cos x	*COS(x)*	–	–	ռադիանով արտահայտված *x* արգումենտի կոսինուսը
arctg x	*ARCTAN(x)*	–	–	ռադիանով արտահայտված *x* արգումենտի արկտանգենսը
ln x	*LN(x)*	–	–	*x* արգումենտի բնական հիմքով լոգարիթմը
e^x	*EXP(x)*	–	–	*e* բնական թվի *x* աստիճանը
$\sqrt{x}$	*SQRT(x)*	–	–	քառակուսի արմատ *x*-ից
x^2	*SQR(x)*	ամբողջ իրական	ամբողջ իրական	*x*-ի քառակուսին
|x|	*ABS(x)*	–	–	*x*-ի բացարձակ արժեքը (մոդուլը)
[x]	*INT(x)*	–	ամբողջ	*x*-ի ամբողջ մասը
{x}	*FRAC(x)*	–	իրական	*x*-ի կոտորակային մասը

π թիվը *Պասկալ* լեզվում նախասահմանված է որպես *pi* անվամբ հաստատուն:

Բնական հիմքով լոգարիթմից բացի, այլ հիմքով լոգարիթմական ֆունկցիայի արժեքը հաշվելու համար անհրաժեշտ է օգտվել լոգարիթմի մի հիմքից մյուսին անցնելու $\log_a b = \frac{\log_c b}{\log_c a}$ բանաձևից՝ *c* հիմքի փոխարեն կիրառելով բնական *e* հիմքը: Այսպիսով՝ $log_a b \rightarrow ln(b)/ln(a)$:

Քանի որ *Պասկալ* լեզվում արգումենտի քառակուսի աստիճանից բարձր աստիճան հաշվող ստանդարտ ֆունկցիա չկա, ապա դրական արգումենտի աստիճանը հաշվելու համար խորհուրդ է տրվում օգտվել մաթեմատիկայից հայտնի $a^n = e^{n \ln(a)}$ առնչությունից:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *COMP* տիպը հիմնականում հարմար է կիրառել հաշվապահական խնդիրներ լուծելիս՝ դրամական հաշվարկներ կատարելու համար:
- Եռանկյունաչափական ֆունկցիաների արգումենտները պետք է տալ ռադիաններով: Աստիճանով արտահայտված անկյունը պետք է վերածել ռադիանի՝ օգտվելով $x_{\text{ռադ.}} = \frac{x_{\text{աստ.}} \cdot \pi}{180}$ բանաձից:
- Որպեսզի կարողանանք օգտվել π-թվին համարժեք *Pi* հաստատունից՝ ծրագիրը պետք է սկսել *USES CRT;* հրահանգով:

1. *Պասկալ* **լեզվում տվյալների ո՞ր տիպերն են անվանում պարզ:**
2. **Թվարկեք ձեզ հայտնի կարգային տիպերը:**
3. **Ինչ արժեքներ են ընդունում տրամաբանական տիպի փոփոխականները:**
4. **Ինչ է սիմվոլային տիպը:**
5. **Ո՞ր տիպերն են կոչվում միջակայքային:**
6. **Եթե** *x=7* **և** *y=5*, **ապա ինչ արժեք կընդունի** *x*-**ը** *DEC(x,y)* **ֆունկցիայի արդյունքում.**
   ա) 2, **բ)** 12:
7. **Եթե** *a=7* **և** *b=2*, **ապա ինչ արժեք կընդունի** *a*-**ն** *INC(a,b)*-**ի արդյունքում.**
   ա) 10, **բ)** 14, **գ)** 6, **դ) որոշված չէ:**
8. **Եթե** *a=13*, **ապա ո՞րն է ճիշտ.**
   ա) *ODD(a)=TRUE*, **բ)** *ODD(a)=FALSE:*

§2.6. ԹՎԱԲԱՆԱԿԱՆ ԵՎ ՏՐԱՄԱԲԱՆԱԿԱՆ ԱՐՏԱՀԱՅՏՈՒԹՅՈՒՆՆԵՐ

Թվաբանական արտահայտությունները կարող են բաղկացած լինել փակագծերի և գործողությունների նշանների միջոցով իրար համակցված հաստատուններից, փոփոխականներից, ստանդարտ ֆունկցիաներից: Կարելի է ասել, որ արտահայտությունները նոր արժեքներ ստանալու միջոցներ են հանդիսանում: Արտահայտությունը, մասնավորապես, կարող է բաղկացած լինել միայն մեկ բաղադրիչից՝ հաստատունից, փոփոխականից կամ ստանդարտ ֆունկցիայի կիրառումից. այս դեպքում արտահայտության արժեքի տիպը համընկնում է տվյալ բաղադրիչի տիպի հետ:

Ընդհանուր դեպքում թվաբանական արտահայտության արժեքի տիպը որոշվում է դրանում ներառված *պարամետրերի տիպերից* **և** *կիրառված գործողություններից:*

Պասկալում որոշված գործողություններից կուսումնասիրենք հետևյալները.

ունար (մեկտեղանի)՝ +, –, *NOT*
մուլտիպլիկատիվ ՝ *, /, *DIV*, *MOD*, *AND*
ադիտիվ (երկտեղանի)՝ +, –, *OR*
հարաբերման գործողություններ՝ =, <, >, >=, <=, <>:

Ունար + և – գործողությունները կիրառվում են արգումենտի նշանի իմաստով (դրական, բացասական), ընդ որում՝ դրական արգումենտի նշանը կարելի է չտալ: Արտահայտության արժեքը հաշվելիս կիրառված գործողությունների կատարման առաջնահերթությունը նվազում է ըստ վերը բերված հաջորդականության. այսպիսով, իրագործման առումով ամենաառաջնահերթն ունար գործողություններն են, իսկ հարաբերման գործողություններն իրականացվում են ամենավերջին հերթին:

Նշենք, որ փակագծերի միջոցով կարելի է անհրաժեշտության դեպքում գործողությունների կատարման առաջնահերթությունը փոխել: Եթե արտահայտության մեջ ներառված գործողությունները կատարման միևնույն առաջնահերթությունն ունեն, ապա իրագործվում են հաջորդաբար՝ ձախից աջ:

Օրինակ՝ *x + y/2* արտահայտության արժեքը հաշվելիս նախ *y*-ը կբաժանվի *2*-ի վրա, ապա ստացվածը կավելացվի *x*-ին: Իսկ *(x + y)/2*-ի դեպքում նախ կիրագործվի *x + y* գումարի հաշվարկը, ապա արդյունքը կբաժանվի *2*-ի վրա: Իսկ *x * y/2* արտահայտության արժեքը հաշվելու համար գործողությունները կիրականացվեն ձախից աջ. նախ կբազմապատկվեն *x* և *y*

փոփոխականների արժեքները, ապա ստացված արտադրյալը կբաժանվի *2*-ի (*Պասկալում* *-ը կիրառվում է որպես բազմապատկման նշան):

Այժմ տանք վերը թվարկած գործողությունների համառոտ բացատրությունները:

Ունար.

`+` դրական թիվ,

`-` բացասական թիվ,

`NOT` *տրամաբանական բացասում*, ընդ որում՝ *NOT(TRUE)=FALSE*, իսկ *NOT(FALSE)=TRUE:*

Մուլտիպլիկատիվ.

`*` բազմապատկում (եթե մասնակից օպերանդներից որևէ մեկն իրական է, ապա արդյունքն իրական թիվ է),

`/` բաժանում (օպերանդների տիպից անկախ՝ արդյունքն իրական թիվ է),

`DIV` ամբողջ տիպի օպերանդների բաժանում (արդյունքը նույնպես ամբողջ է),

`MOD` ամբողջ տիպի օպերանդների բաժանման արդյունքի ամբողջ մնացորդի ստացում,

`AND` տրամաբանական ԵՎ (արդյունքը հավասար է *TRUE* միայն այն դեպքում, երբ բոլոր մասնակից օպերանդները *TRUE* արժեք ունեն):

Ադիտիվ.

`+` գումարում (արդյունքն իրական է, եթե գումարելիներից մեկն իրական է),

`-` հանում (արդյունքն իրական է, եթե օպերանդներից մեկն իրական է),

`OR` տրամաբանական ԿԱՄ (արդյունքը *TRUE* է, եթե գործողությանը մասնակցող օպերանդներից որևէ մեկն ունի *TRUE* արժեք):

Հարաբերման գործողություններ.

`=` ստուգում է օպերանդների հավասարությունը. արդյունքը *TRUE* է, եթե օպերանդները հավասար են, հակառակ դեպքում՝ *FALSE*,

`<` եթե անհավասարման ձախ մասի արժեքը փոքր է աջ մասի արժեքից՝ վերադարձվում է *TRUE*, հակառակ դեպքում՝ *FALSE*,

`>` եթե անհավասարման ձախ մասի արժեքը մեծ է աջ մասի արժեքից՝ վերադարձվում է *TRUE*, հակառակ դեպքում՝ *FALSE*,

`<>` անհավասարման ստուգում. արդյունքը *TRUE* է, եթե օպերանդների արժեքներն իրար հավասար չեն,

`<=` եթե անհավասարման ձախ մասի արժեքը փոքր է կամ հավասար աջ մասի արժեքից՝ վերադարձվում է *TRUE*, հակառակ դեպքում՝ *FALSE*,

`>=` եթե անհավասարման ձախ մասի արժեքը մեծ է կամ հավասար աջ մասի արժեքից՝ վերադարձվում է *TRUE*, հակառակ դեպքում՝ *FALSE:*

Հարաբերման **կամ** *NOT, AND, OR* **գործողություններ պարունակող արտահայտությունները կոչվում են** *տրամաբանական արտահայտություններ* **և տրամաբանական (***TRUE, FALSE***) արժեք ունեն:**

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ *NOT, AND, OR* տրամաբանական գործողությունները կիրառելի են նաև ամբողջ տիպի օպերանդների համար. այս դեպքում արդյունքը նույնպես ամբողջ թիվ է (գործողություններն իրականացվում են օպերանդների համապատասխան բիթերի հետ՝ ըստ 2.4 աղյուսակի):

Աղյուսակ 2.4

Տրամաբանական (կարգային) գործողություններ ամբողջ տիպի հետ				
I օպերանդ	II օպերանդ	NOT	AND	OR
1	–	0	–	–
0	–	1	–	–
0	0	–	0	0
0	1	–	0	1
1	0	–	0	1
1	1	–	1	1

1. *2cosx+0.1sinx²* **արտահայտությունը հանրահաշվական, թե՞ տրամաբանական արտահայտություն է:**
2. **Եթե** *x=2*, **ապա ի՞նչ արժեքներ կընդունեն հետևյալ արտահայտությունները.**
   ա) *(x>1) AND (x<7)*, **բ)** *(x>0) OR (x<-5)*,
   գ) *NOT (x>1)*, **դ)** *7 DIV x*, **ե)** *10 MOD x*:
3. **Ստորև բերված արտահայտությունները գրեք** *Պասկալ* **լեզվով.**

ա) $x^3+8\sin x\cos 3x+\log_3 4x^2$ **բ)** $\frac{x^2-4}{y^2+2}+2^{\sin x}$ **գ)** $\ln\left(e^x+1\right)+\sqrt{\left(x^2+4\right)}$

դ) $\sin\left(\frac{3x+4}{y+2}\right)+\sqrt{(x+3)^3}$ **ե)** $\sqrt[4]{x^2+3}+\left(|x|+2\right)^2$

§2.7. ՄԵԿՆԱԲԱՆՈՒԹՅՈՒՆՆԵՐ: ՎԵՐԱԳՐՄԱՆ ՕՊԵՐԱՏՈՐ: ՆԵՐՄՈՒԾՄԱՆ ՀՐԱՀԱՆԳ

Ծրագիրն առավել ընթեռնելի դարձնելու նպատակով հատուկ միջոցներ՝ *մեկնաբանություններ* են կիրառվում: *Պասկալում* մեկնաբանությունը պայմանանշանների ցանկացած հաջորդականություն է, որն առնվում է ձևավոր *{ }* փակագծերի կամ *(* *)* նշանների միջև: Օրինակ՝

```
{Սա Պասկալ լեզվի մեկնաբանություն է:}     կամ
(*Ծրագիրը հաշվում է առաջին 100 պարզ թվերի գումարը*):
```

Մեկնաբանություն կարելի է տալ ծրագրի ցանկացած մասում:

Վերագրման օպերատորը ծառայում է փոփոխականին արժեք տալու համար: Ընդհանուր դեպքում այն կարելի է ներկայացնել հետևյալ կերպ՝

```
A:=B;
```

որտեղ *A*-ն վերագրման արդյունքում արժեք ստացող փոփոխականն է, իսկ *B*-ն՝ ցանկացած արտահայտություն, որի արժեքը պետք է լինի նույն տիպի, ինչ *A* փոփոխականինը:

Վերագրման օպերատորի մեջ աջ մասի արտահայտության արժեքը պետք է լինի ձախ մասի փոփոխականի տիպի (բացառություն է կազմում այն դեպքը, երբ փոփոխականն իրական տիպի է, իսկ արտահայտության արժեքը՝ ամբողջ, բայց ոչ հակառակը):

Պետք է հիշել, որ *վերագրման արդյունքում A փոփոխականի ունեցած նախկին արժեքը ոչնչանում է*: Նկատենք նաև, որ վերագրման օպերատորում պարտադիր կերպով մասնակցող *:=* պայմանանշանների համակցությունը գրառվում է իրար կից՝ առանց դրանց միջև բացատանիշի: Օրինակ՝

```
x:= 2; {1}
x:= 5*(4+x);          {2}
```

Եթե բերվածը դիտարկենք որպես իրար հաջորդող հրամաններ, ապա ըստ {1} տողի՝ *x* փոփոխականը ստանում է 2 արժեքը: {2} տողի արդյունքում համակարգիչը նախ *x*-ի ընթացիկ արժեքը (2) տեղադրելով աջ մասի արտահայտության մեջ՝ կհաշվի *4 + x*-ի արժեքը (6), ապա ստացվածը բազմապատկելով *5*-ով՝ արդյունքը կվերագրի *x*-ին: Այսպիսով, արդյունքում *x*-ը նախկին (2) արժեքի փոխարեն կստանա *30* արժեքը:

Ծրագրի կատարման ընթացքում հաճախ անհրաժեշտ է լինում փոփոխականներին արժեքներ տալ ստեղնաշարից: Այս գործընթացն անվանում են *ընթացիկ տվյալների ներմուծում*: Ստեղնաշարից տվյալներ ներմուծելու

նպատակով *Պասկալում* կիրառում են *READ* և *READLN* հրահանգները (հետագայում՝ հրաման), որոնք կարելի է նկարագրել հետևյալ կերպ.

READ (մուտքի փոփոխականների ցուցակ);
READLN (մուտքի փոփոխականների ցուցակ);

Այս հրամաններում փակագծերի մեջ ստորակետերով անջատելով թվարկում են այն փոփոխականները, որոնց արժեքները պետք է տալ ստեղնաշարից: Օրինակ՝

```
READ(A,B,C);          {1}    կամ՝
READLN(A,B,C);        {2}
```

Համակարգիչը հաջորդաբար կատարելով ծրագրում ներառված հրամանները՝ ներմուծման հրամանն իրականացնելիս ընդհատում է աշխատանքը և ստեղնաշարից, մուտքի ցուցակում ներառված փոփոխականների քանակին համապատասխան, արժեքներ ներմուծելուն սպասում: Ընդ որում, ներմուծված արժեքները հաջորդաբար վերագրվում են մուտքի ցուցակում թվարկված փոփոխականներին: Տվյալները կարելի է ներմուծել միևնույն տողում՝ դրանք իրարից բացատանիշերով անջատելով և ներմուծման գործընթացն ավարտելով *ENTER* ստեղնով, կամ տարբեր տողերում, ամեն արժեք ներմուծելուց հետո սեղմելով *ENTER* ստեղնը:

READ և *READLN* հրամանների տարբերությունը տեսնելու համար շարունակենք քննարկել վերը բերված օրինակը: Եթե *A, B* և *C* փոփոխականների արժեքները ներմուծվում են *READ* հրամանով, ապա տվյալները կարելի է ներմուծել հետևյալ եղանակներից ցանկացածով.

```
ա) 5  -7  8     (ENTER),
բ) 5  -7  (ENTER),
   8  (ENTER),
գ) 5  (ENTER),
  -7  (ENTER),
   8  (ENTER):
```

Բերված բոլոր դեպքերում *A* փոփոխականը կստանա *5*, *B*-ն՝ *-7*, իսկ *C*-ն՝ *8* արժեքը:

Այժմ քննարկենք *READLN*-ի կիրառման դեպքը. եթե ներմուծումը կատարվի ա) եղանակով, ապա արդյունքը համարժեք կլինի *READ*-ի արդյունքին, իսկ բ) և գ) դեպքերն այլ արդյունք կտան: Բանն այն է, որ *READLN* հրամանն *ENTER*-ը համարում է ներմուծման գործընթացի ավարտ, և այդ պատճառով բ) դեպքում *A* և *B* փոփոխականներն արժեքներ կստանան. *C*-ն՝ ոչ, իսկ գ) դեպքում արժեք կստանա միայն *A*-ն:

Եթե, օրինակ, փորձենք *A, B, C* փոփոխականների արժեքները ներմուծել

```
READLN(A,B);      {1}
READ(C);          {2}
```

հրամաններով, և արժեքները ներմուծենք միևնույն տողի վրա, հետևյալ կերպ՝

5 -7 8 (ENTER),

ապա կներմուծվեն միայն *A*-ն և *B*-ն, իսկ *C*-ի համար նախատեսված *8* արժեքը կանտեսվի: Համակարգիչն ըստ {2} հրամանի կսպասի ևս մեկ արժեքի ներմուծման, որն էլ կվերագրի *C*-ին:

READLN հրամանն ունի կիրառման ևս մեկ եղանակ՝ առանց մուտքի ցուցակի.

READLN;

Այս դեպքում համակարգիչն ընդհատում է ծրագրի ընթացքը, սպասում ստեղնաշարի ցանկացած ստեղն սեղմելուն, որից հետո շարունակում է աշխատանքը:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *READ* և *READLN* հրամաններն իրականում ստանդարտ *ենթածրագրեր* են՝ առավել լայն գործածման հնարավորություններով:

1. *Պասկալում* **մեկնաբանություններ ստեղծելու քանի՞ եղանակ գիտեք:**
2. **Ո՞րն է ծրագրում մեկնաբանություններ տալու նպատակը:**
3. **Ո՞ր գրառումներն են ճիշտ վերագրումներ.**
   ա) *a:=5;* **բ)** *7:=C;* **գ)** *d:=5 * sin(x);*
4. **Եթե** *x*-**ը իրական փոփոխական է, ապա ո՞ր գրառումներն են ճիշտ.**
   ա) *x:=3 * ln(5);* **բ)** *X:=-2;* **գ)** *X:='C';* **դ)** *x:=true;*
5. **Քանի՞ ներմուծման հրաման գիտեք:**
6. **Ո՞րն է** *READ* **և** *READLN* **հրամանների տարբերությունը:**

§2.8. ԱՐՏԱԾՄԱՆ ՀՐԱՀԱՆԳ: ԱՐՏԱԾՄԱՆ ՁԵՎԱՉԱՓ

Ծրագրի աշխատանքի ընթացքում ստացված արդյունքներն էկրանին արտածելու համար կիրառում են *WRITE* և *WRITELN* հրահանգները (հետագայում՝ հրաման), որոնց ընդհանուր տեսքը հետևյալն է.

WRITE(ելքի ցուցակ);
WRITELN(ելքի ցուցակ);

որտեղ ելքի ցուցակը կարող է պարունակել հաստատուն արժեքներ, փոփոխականներ, ապաթարցերի (‘ ‘) մեջ առնված տեքստային ինֆորմացիա, տրամաբանական և թվաբանական արտահայտություններ:

WRITE և *WRITELN* հրամանների աշխատանքի տարբերությունը տեսնելու համար քննարկենք հետևյալ օրինակը.

```
WRITE (4); WRITE(5); WRITE(6);     {1}
WRITELN(4,5);                      {2}
WRITE(6);                          {3}
```

{1} դեպքում մեկ տողի վրա, իրար կից, կարտածվեն ելքի ցուցակներում ներառված *4, 5* և *6* թվերը և արդյունքում էկրանին կտեսնենք *456* թիվը:

{2} դեպքում մեկ տողի վրա կարտածվեն միայն *4*-ն ու *5*-ը՝ կազմելով *45*, իսկ *6*-ը կարտածվի հաջորդ տողում (ըստ {3} հրամանի):

Այսպիսով, *WRITELN*-ը, ի տարբերություն *WRITE*-ի, ելքի ցուցակում ներառվածն արտածելուց հետո տողադարձ է կատարում: Արտածման հրամաններում ապաթարցերի մեջ ներառված ցանկացած ինֆորմացիա արտածվում է անփոփոխ. օրինակ՝ *WRITE*(*‘Ես աշակերտ եմ’*); հրամանով էկրանին կբերվի *Ես աշակերտ եմ,* իսկ *WRITELN(‘a=’,5);* հրամանով՝ *a=5* հաղորդագրությունը:

Արտածվող տվյալները էկրանին առավել ցանկալի, ընթեռնելի տեսքով ներկայացնելու նպատակով կարելի է տվյալների արտածման գործընթացը ծրագրային միջոցներով ղեկավարել:

WRITE և *WRITELN* հրամանների միջոցով արտածվող յուրաքանչյուր բաղադրիչին հաջորդող *:* (վերջակետ) նշանից հետո կարելի է նշել համապատասխան տվյալն արտածելու համար տրամադրվող դիրքերի ցանկալի քանակը (դաշտի լայնության չափը) հետևյալ կերպ.

WRITE(x:k);

որտեղ *x*-ը արտածվող փոփոխականն է, *k*-ն արտածման համար էկրանին տրամադրվող դիրքերի քանակը: Եթե *x*-ը ամբողջ տիպի փոփոխական է, որի թվանշանների քանակը (ներառյալ նաև “–“ պայմանանշանը, եթե թիվը բացասական է) տրված *k*-ից քիչ է, ապա *x*-ի արժեքն արտածելիս այն ձախ մասից կլրացվի համապատասխան քանակությամբ բացատանիշերով: Օրինակ, եթե *a=72*, և տրվել է *WRITE(‘a=’, a : 5)*; հրամանը, ապա կարտածվի *a=* ␣␣␣ *72* ինֆորմացիան, որտեղ ␣-ով նշվել է բացատանիշը:

Եթե *k*-ն ավելի փոքր է, քան արտածվող թվի թվանշանների քանակը, ապա *k*-ի մեծությունն անտեսվում է, և թիվն արտածվում է ամբողջությամբ՝ սկսած ընթացիկ դիրքից:

Եթե արտածվող տվյալն իրական տիպի է, ապա արտածման ձևաչափ չնշելու դեպքում արժեքն արտածվում է, այսպես կոչված, *էքսպոնենտային* տեսքով, այն նախապես բերելով *նորմալ տեսքի*, որտեղ տասնորդական կետից ձախ միայն մեկ թվանշան է գրվում: Օրինակ, եթե *a*-ն իրական տիպի է, որի արժեքը *52.15* է, ապա դրա արժեքը կարտածվի *5.2150000000E+01* տեսքով, որն, իհարկե, հարմար չէ ընթերցել: Նման դեպքերում արտածվող ինֆորմացիան առավել ընթեռնելի դարձնելու նպատակով, արտածման դաշտի լայնությունից զատ, հնարավոր է տալ նաև տասնորդական կետից հետո պահանջվող թվանշանների քանակը (թվի ճշտության չափը)՝ հետևյալ կերպ՝ *WRITE(x : k : m)*; : Օրինակ, եթե *a*-ի *52.15* արժեքն արտածելու համար կիրառվեր *WRITE('a=',a:10:2)*; հրամանը, ապա արդյունքում կտեսնեինք *a=␣␣␣␣␣52.15* ինֆորմացիան:

Եթե *WRITE(x : k : m)* հրամանով արտածվող *x*-ի արժեքում տասնորդական կետին հաջորդող մասը *m* հատից ավելի թվանշաններ է պարունակում, ապա արտածվող թիվը կլորացվում է՝ ըստ մաթեմատիկայում ընդունված օրենքների: Օրինակ, եթե *x=101.567*, և այն արտածվում է *WRITE('x=',x:10:2);* հրամանով, ապա կստացվի.

x=␣␣␣␣101.57

Եթե *m*-ի արժեքը մեծ է թվի տասնորդական մասի թվանշանների քանակից, ապա արտածվելիս թվի վերջում համապատասխան քանակությամբ *0*-ներ են կցագրվում: Օրինակ, եթե նախորդ դեպքում տրված լիներ *WRITE('x=',x:10:4)*; հրամանը, ապա կստանայինք *x =␣␣101.5670* պատասխանը:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *WRITE, WRITELN* հրամաններն իրականում ստանդարտ ենթածրագրեր են՝ առավել լայն գործածման հնարավորություններով:
- Տվյալների ներմուծման գործընթացն առավել հասկանալի դարձնելու նպատակով յուրաքանչյուր ներմուծվող փոփոխականի համար կարելի է նախ կիրառել *WRITE* հրամանը՝ ելքի ցուցակում ապաթարցերի մեջ տվյալ փոփոխականի վերաբերյալ ինֆորմացիայով: Օրինակ՝

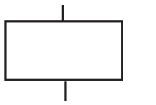
```
WRITE('x='); READ(x);
WRITE('y='); READ(y);
```

1. **Ո՞րն է** *WRITE* **և** *WRITELN* **հրամանների տարբերությունը:**
2. *x*-**ը ամբողջ փոփոխական է, որի արժեքը** *215* **է: Ինչ կարտածվի** *WRITE('x=',x:2)* **հրամանով.**
   ա) *x=21*, **բ)** *x=15*, **գ)** *x=215:*
3. *y*-**ը իրական փոփոխական է, որի արժեքը** *-16.127* **է: Ինչ կարտածվի** *WRITE('y=',y:4:2)*; **հրամանով.**
   ա) *x=-16*, **բ)** *x=-16.1*, **գ)** *x=-16.12*, **դ)** *x=-16.13*, **ե)** *y=-16.127:*

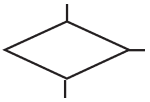
§2.9. ԳԾԱՅԻՆ ԱԼԳՈՐԻԹՄՆԵՐԻ ԾՐԱԳՐԱՎՈՐՈՒՄ

9-րդ դասարանի դասընթացից ծանոթ եք *ալգորիթմ* հասկացությանն ու *ալգորիթմ նկարագրելու* եղանակներին: Վերհիշենք ալգորիթմներից հայտնի նյութը:

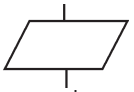
Ալգորիթմը քայլերի (գործողությունների) կարգավորված հաջորդականություն է, որը հանգեցնում է սպասված արդյունքին: *Ալգորիթմ* նկարագրելու եղանակներից ծանոթ եք բառաբանաձևային և գրաֆիկական եղանակներին: Քանի որ օգտվելու ենք ալգորիթմ նկարագրելու գրաֆիկական եղանակից՝ վերհիշենք դրանում կիրառվող բլոկների նշանակությունները.



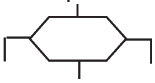
հաշվարկների կատարման և վերագրման գործողություն,



պայմանի ստուգում և հաշվման գործընթացի այլընտրանքային շարունակում,

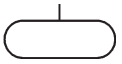


տվյալների ներմուծում, տվյալների արտածում,



ցիկլային գործընթացի կազմակերպում,

ալգորիթմի սկիզբ,



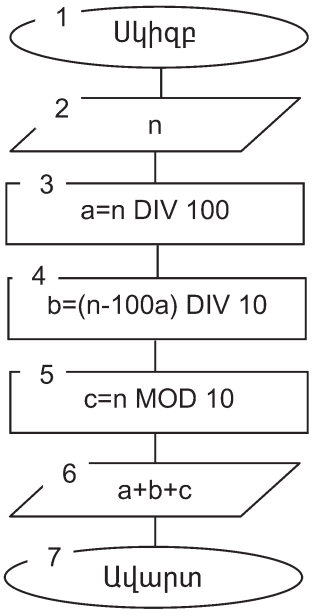
ալգորիթմի ավարտ,

ալգորիթմի հոսքի ընդհատված մասերի միջև կապի միջոց:

Ալգորիթմները, կախված տվյալ պահին լուծվող խնդրից, կարող են լինել *գծային*, *ճյուղավորված* և *ցիկլային*: Գծային են այն ալգորիթմները, որ-

տեղ, պարամետրերի արժեքներից անկախ, գործողությունները կատարվում են միշտ միևնույն հաջորդականությամբ՝ վերից վար, յուրաքանչյուրը՝ միայն մեկ անգամ: Դիտարկենք հետևյալ խնդիրը՝ հաշվել ներմուծված *n* եռանիշ թիվը կազմող բաղադրիչ թվանշանների գումարը:

Նախ կազմենք խնդրի լուծման բլոկ-սխեման (նկ. 2.3.), ապա՝ ծրագիրը:



Նկ.2.3. Եռանիշ թվի թվանշանների գումարի հաշվման ալգորիթմ

Բերված ալգորիթմում կիրառվել են ամբողջ թվերի համար սահմանված *DIV* և *MOD* ստանդարտ ֆունկցիաները, որտեղ *a DIV b* -ն վերադարձնում է *a*-ն *b*-ի բաժանելիս ստացվող ամբողջ արժեքը (օրինակ՝ *7 DIV 3=2*), իսկ *MOD*-ը՝ այդ բաժանման ամբողջ մնացորդը (օրինակ՝ *7 MOD 3=1*):

Եթե, օրինակ՝ *n=672*, ապա *3*-րդ բլոկով կստանանք՝ $a = 672 \text{ DIV } 100 = 6$, *4*-րդ բլոկով՝ $b = (672 - 100 \cdot 6) \text{ DIV } 10 = 7$, իսկ *5*-րդով՝ $c = 672 \text{ MOD } 10 = 2$: Այսպիսով՝ *a, b, c* փոփոխականներում ստացվել են եռանիշ թվի բաղադրիչները, մնում է *6*-րդ բլոկով արտածել պահանջվող գումարը:

Կազմենք ծրագիրը.

```
PROGRAM Eranish;
VAR n:Word;   {Սա եռանիշ թվի համար է:}
a,b,c: BYTE;  {a-ն հարյուրավորի, b-ն տասնավորի, c-ն միավորի համար է}
    BEGIN WRITE('n='); READ(n); {եռանիշ թվի ներմուծում}
          a:=n DIV 100;  {հարյուրավորի ստացում}
          b:=(n-100*a) DIV 10;  {տասնավորի ստացում}
          c:=n MOD 10;  {միավորի ստացում}
```

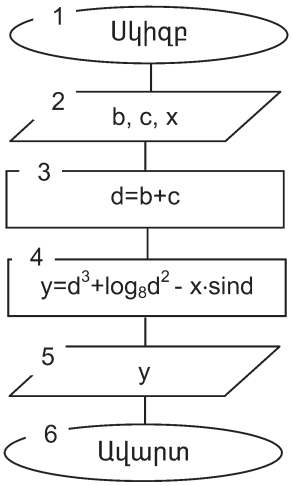
```
WRITELN( 'n-ի թվանշանների գումարը =',a+b+c:4)
END.
```

Ծրագրում *n*-ը հայտարարված է որպես *WORD* տիպի փոփոխական, քանի որ պետք է պարունակի դրական ամբողջ թիվ, բայց չի կարող բնութագրվել որպես *BYTE*, քանի որ *BYTE*-ում տեղավորվող ամենամեծ թիվը *255*-ն է (իսկ եռանիշի վերին եզրը հավասար է *999*-ի): *a,b,c* փոփոխականների արժեքները նույնպես դրական ամբողջ թվեր են, որոնց արժեքները չեն կարող *9*-ից մեծ լինել. նման արժեքների համար առավել հարմար է *BYTE* տիպը:

Նկատեք, որ ծրագրում բոլոր հրամաններն ավարտվել են կետ-ստորակետերով (;), բացի *END*-ին նախորդող հրամանից: Պատճառն այն է, որ *END*-ը ոչ միայն սահմանափակում է *BEGIN*-ով սկսված բլոկը, այլև ավարտում իրեն նախորդող օպերատորը: Այսպիսով, կետ-ստորակետն այս դեպքում կլիներ ավելորդ, և *Պասկալի* կոմպիլյատորը չնայած որևէ սխալ չէր գտնի, սակայն «կմտածեր», թե *END*-ի և վերջին օպերատորի միջև օպերատոր կա: Նման՝ հրաման չպարունակող օպերատորն անվանում են *դատարկ օպերատոր*: Դատարկ օպերատոր կա նաև երկու իրար հաջորդող կետ-ստորակետերի միջև: Օրինակ *x:=8; ; y:=-7;* :

Ինչպես երևում է վերը բերված ծրագրից՝ գծային ալգորիթմների օգնությամբ լուծվող հաշվարկային խնդիրները կարող են պարունակել միայն ներմուծման, արտածման հրամաններ և հաշվարկներ կատարելու համար՝ վերագրման օպերատորներ:

Դիտարկենք գծային ալգորիթմով լուծվող ևս մի խնդիր. *x, b, c* պարամետրերի ցանկացած իրական արժեքների համար հաշվել և արտածել *y*-ի արժեքը, եթե $y = (b + c)^3 = \log_8 (b + c)^2 - x \sin (b + c)$: Նախ կառուցենք խնդրի լուծման բլոկ-սխեման (նկ. 2.4.).



Նկ.2.4. $y = (b+c)^3 + \log_8(b+c)^2 - x \sin(b+c)$ *արտահայտության հաշվման ալգորիթմ*

3-րդ բլոկում մտցված լրացուցիչ *d* փոփոխականի մեջ պահվել է *b+c* արտահայտության արժեքը, որպեսզի *4*-րդ բլոկում ներառված արտահայտության արժեքը հաշվարկելիս *b+c*-ի արժեքը մի քանի անգամ չհաշվենք: Գրենք ծրագիրը.

```
PROGRAM Hashvark;
VAR d,b,c,x,y:REAL;                                                    {1}
    BEGIN  WRITE('b='); READ(b); WRITE('c='); READ(c); WRITE('x='); READ(x);
    d:=b+c;
    y:=EXP(3*LN(d))+LN(SQR(d))/ LN(8)-x*SIN(d) ;                       {2}
    WRITELN('y=',y :8 :3)
END.
```

{2} տողում կատարված հաշվարկի մեջ d^3-ը հաշվվել է *EXP(3*LN(d))* բանաձևով՝ ըստ $a^b = e^{b\cdot \ln a}$ մաթեմատիկայից հայտնի առնչության, իսկ $\log_8 d^2$-ն հաշվվել է լոգարիթմի մի հիմքից այլ հիմքին անցնելու կանոնով (քանի որ *Պասկալում* հայտնի է միայն բնական հիմքով լոգարիթմը) և *8* հիմքով լոգարիթմից անցում է կատարվել բնական հիմքի.

$$\log_8 d^2 = \frac{\ln d^2}{\ln 8} = \ln(sqr(d))/\ln(8) :$$

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Պահանջվող տվյալների ճիշտ հայտարարությունը ծրագիր կազմելիս կարևորվում է, քանի որ համակարգչի օպերատիվ հիշողությունն անհրաժեշտ է խնայողաբար օգտագործել:

1. **Գծային ալգորիթմներում ո՞ր հրամաններն ու օպերատորներն են կիրառվում:**
2. **Ի՞նչ է դատարկ օպերատորը, ինչպե՞ս է այն կազմվում:**
3. **Հաշվել և արտածել տրված քառանիշ թվի թվանշանների արտադրյալը:**
4. **Տրված *x* և *a* իրական ցանկացած արժեքների համար հաշվել և արտածել *y*-ի արժեքը, եթե.**

ա) $y = (x + 1)(x^2 + 1)\sin(x + 3)$ **բ)** $y = \sqrt[3]{x+2} + \frac{x+2}{x^2+6}$:

§2.10. ՃՅՈՒՂԱՎՈՐՄԱՆ ԳՈՐԾԸՆԹԱՑԸ ԱԼԳՈՐԻԹՄՆԵՐՈՒՄ: ՃՅՈՒՂԱՎՈՐՄԱՆ (ՊԱՅՄԱՆԻ) ՕՊԵՐԱՏՈՐ

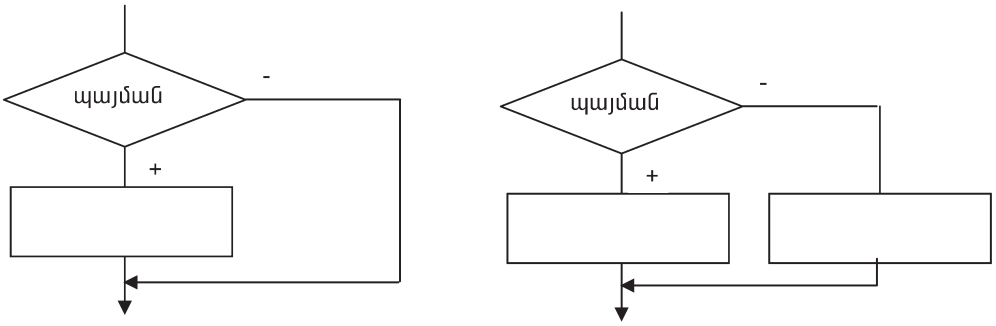
Խնդիրների լուծման ալգորիթմները, ի տարբերություն գծայինի, հաճախ *ճյուղավորումներ* են պարունակում: Պատճառը լուծման մեջ առկա պայմաններն են, որոնցից կախված խնդրի հետագա լուծումը շարունակվում է տարբեր ճանապարհներով: Հիշենք, որ ալգորիթմը, որտեղ, ստուգվող պայմանից կախված խնդրի լուծման գործընթացը շարունակվում է տարբեր ուղիներով, անվանում են *ճյուղավորված*, իսկ տվյալ ուղիները՝ *ճյուղեր*:

Ճյուղավորումներ պարունակող ալգորիթմները ծրագրավորելու նպատակով *պայմանի օպերատոր* է նախատեսված, որն ունի հետևյալ ընդհանուր տեսքը.

IF a THEN b ELSE c;

Այստեղ *a*-ն տրամաբանական արտահայտություն է, *b*-ն և *c*-ն՝ ցանկացած օպերատոր կամ *BEGIN* ու *END* բառերի միջև առնված օպերատորների համախմբություն՝ բլոկ:

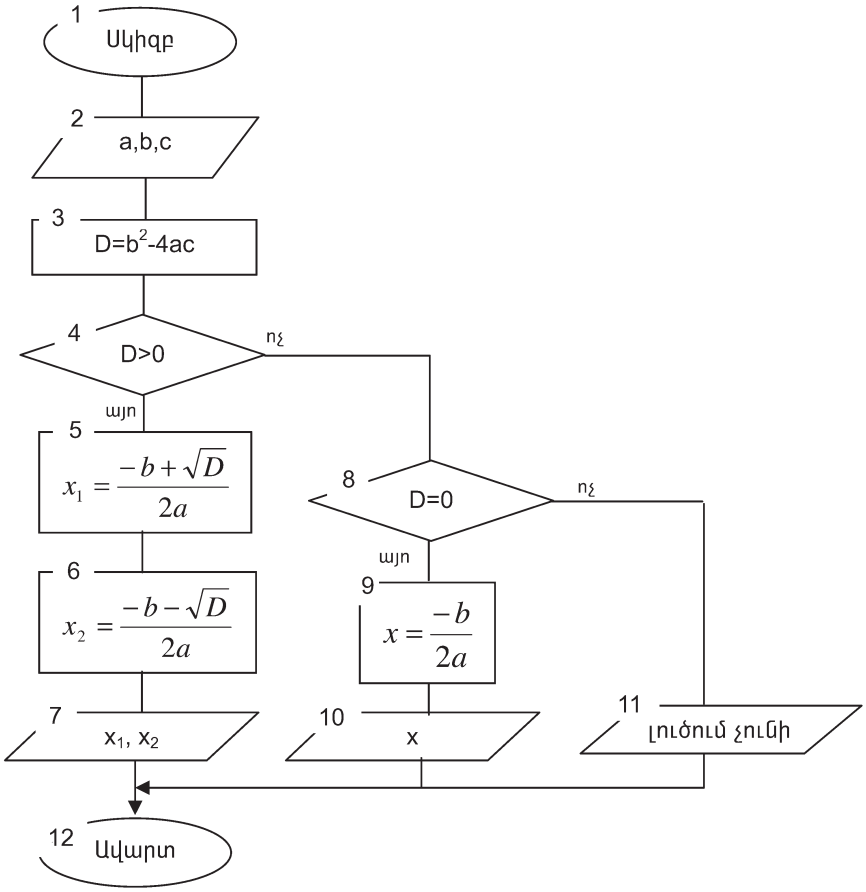
Եթե պայմանի օպերատորը ծրագրավորում է նկ. 2.5. ա)-ում բերված տիպի գործընթաց, ապա կիրառվում է *պայմանի օպերատորի համառոտ* տեսքը՝ *IF a THEN b*; , իսկ նկ. 2.5. բ)-ում բերվածի դեպքում՝ *IF a THEN b ELSE c*; տեսքի պայմանի օպերատորը, որն անվանում են *ընդարձակ*:



ա) *բ)*

Նկ. 2.5. Ալգորիթմների ճյուղավորումը

Որպես օրինակ դիտարկենք $ax^2 + bx + c = 0$, ($a \neq 0$) քառակուսի հավասարման արմատները գտնելու ալգորիթմը (նկ. 2.6.):



Նկ. 2.6. Քառակուսի հավասարման արմատները փնտրելու ալգորիթմ

Բլոկ-սխեմայից երևում է, որ *4*-րդ բլոկում ներառված պայմանի ճշմարիտ (*TRUE*) կամ կեղծ (*FALSE*) լինելուց կախված՝ խնդրի լուծումը շարունակվում է տարբեր ուղղություններով, ընդ որում, պայմանի կեղծ լինելու դեպքում ըստ *8*-րդ բլոկում ներառված պայմանի ընդունած արժեքի՝ ալգորիթմի մեջ մեկ այլ ճյուղավորում է առաջանում: Կազմենք համապատասխան ծրագիրը.

```
PROGRAM Qar_Havasarum;
VAR a,b,c,z,d:REAL;
      x1,x2,x:REAL;
BEGIN  WRITE('a='); READ(a);
      WRITE('b='); READ(b);
      WRITE('c='); READ(c);
      d:=SQR(b)-4*a* c;                                    {1}
```

```
        z:=2* a;                                                    {2}
        IF d>0 THEN
                BEGIN
                    x1:=(-b-SQRT(d))/z;  x2:=(-b+SQRT(d))/z;
                    WRITELN('x1=',x1:10:2,'      ':3, 'x2=',x2:10:2)    {3}
                END
                    ELSE IF d=0 THEN
                   BEGIN
                      x:=-b/z;
                     WRITELN('հավասարումը մեկ արմատ ունի՝ x=',x:10:2)
                   END
                     ELSE WRITELN('հավասարումը լուծում չունի')
END.
```

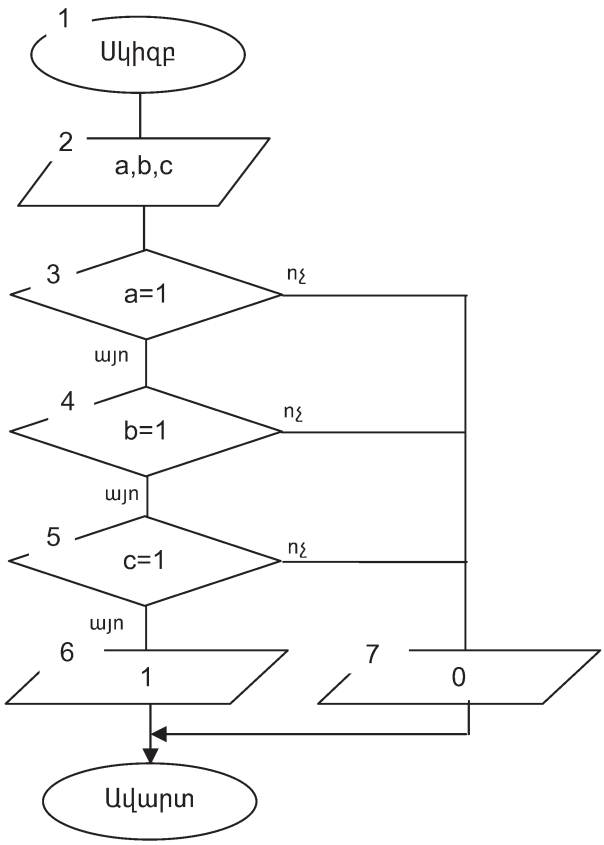
Ծրագրի {1} տողում հաշվարկվել ու *d*-ի մեջ պահպանվել է քառակուսի հավասարման որոշիչը, իսկ {2} տողում z-ին վերագրվել է *2*a*-ի արժեքը՝ չնայած բլոկ-սխեմայում դրա անհրաժեշտությունը չէր զգացվում: Բանն այն է, որ ծրագրում անհրաժեշտ է եղել *2*a*-ի արժեքը կիրառել մի քանի անգամ, և մեկ անգամ հաշվելով ու պահպանելով՝ ծրագրի արագագործությունը դրանից միայն կշահի:

Դիտելով նկ. 2.6.-ում բերված բլոկ-սխեման՝ նկատենք, որ *4*-րդ և *8*-րդ բլոկներում ներառված պայմաններից յուրաքանչյուրի ճշմարիտ լինելու դեպքում մեկից ավելի հրամաններ պետք է իրագործվեն: Նման դեպքերում այդ հրամաններից բլոկ է կազմվում:

{3} մեկնաբանությամբ տողում կիրառված *WRITELN* հրամանով *x1* և *x2* արմատների արժեքներն արտածվում են միևնույն տողի վրա. որպեսզի իրարից փոխանջատվեն՝ դրանց միջև մտցվել է '‗' : *3* պարամետրը, ըստ որի *x1*-ի և *x2*-ի արժեքներն իրարից կտարանջատվեն *3* իրար կից բացատանիշերով:

Քննարկենք հետևյալ խնդիրը. եթե ներմուծված *a, b* և *c* ամբողջ փոփոխականների արժեքները հավասար են *1*-ի՝ արտածել *1*, հակառակ դեպքում (եթե դրանցից թեկուզ մեկը հավասար չէ *1*-ի) արտածել *0* թիվը: Կառուցենք խնդրի լուծման բլոկ-սխեման և ծրագիրը.

```
PROGRAM Baxadryal;
VAR a,b,c:BYTE;
BEGIN   WRITE('a='); READ(a);
   WRITE('b='); READ(b);
   WRITE('c='); READ(c);
   IF (a=1)AND(b=1)AND(c=1) THEN WRITELN(1)
       ELSE WRITELN(0)
END.
```

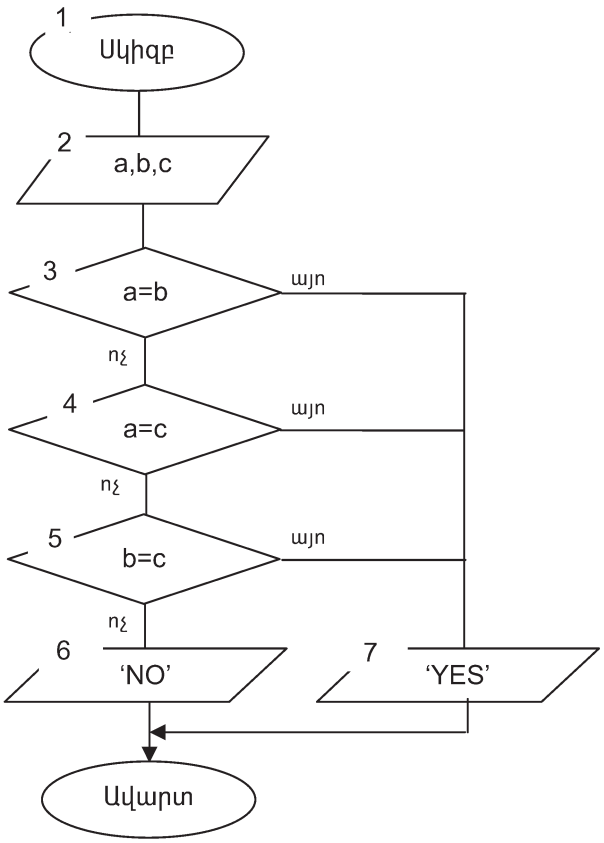


Նկ.2.7. Ճյուղավորված ալգորիթմով խնդրի լուծման օրինակ

Փորձենք խնդրի պահանջը ձևակերպել մեկ այլ եղանակով: Եթե միաժամանակ ճիշտ են *a=1* և *b=1* և *c=1* պայմանները, այլ կերպ ասած՝ եթե նշված *պայմանները համակարգելի են*, արտածել *1*, հակառակ դեպքում՝ *0*: *Համակարգելի պայմանները* բլոկ-սխեմայում իրար կցվում են պայմանների ճշմարիտ (այո) ճյուղերի ուղղություններով, իսկ ծրագրում՝ *AND* առանցքային բառի միջոցով:

AND-ի միջոցով կազմավորված բաղադրյալ պայմանը *ճշմարիտ է*, եթե ճշմարիտ են բոլոր բաղադրիչ պայմանները, և դրանցից թեկուզ մեկի կեղծ լինելու դեպքում *կեղծ է* ամբողջ բաղադրյալ պայմանը:

Այժմ լուծենք հետևյալ խնդիրը. արտածել *YES*՝ եթե *a, b, c* իրական թվերի մեջ գոյություն ունեն իրար հավասար թվանշաններ, հակառակ դեպքում՝ *NO* հաղորդագրությունը: Նախ կազմենք խնդրի լուծման բլոկ-սխեման.



Նկ. 2.8. Ճյուղավորված ալգորիթմի օրինակ

Այս խնդրի պահանջը ևս փորձենք այլ կերպ ձևակերպել: Եթե ճշմարիտ է *a=b կամ a=c կամ b=c* պայմաններից գոնե մեկը, այլ կերպ ասած՝ եթե ունենք *համախմբելի պայմաններ*, ապա արտածել *YES*, հակառակ դեպքում՝ *NO*: Համախմբելի պայմանները բլոկ-սխեմայում իրար կցվում են պայմանների կեղծ (ոչ) ճյուղերի ուղղություններով, իսկ ծրագրում՝ *OR* առանցքային բառի միջոցով:

> ***OR*-ի միջոցով կազմավորված բաղադրյալ պայմանը** *ճշմարիտ է*, **եթե ճշմարիտ է այն կազմող բաղադրիչ պարզ պայմաններից գոնե մեկը**:

Կազմենք նկ. 2.8.-ում բերված ալգորիթմին համապատասխանող ծրագիրը.

```
PROGRAM Yes_No;
VAR a,b,c:REAL;
BEGIN      WRITE('a='); READ(a);
    WRITE('b='); READ(b);
    WRITE('c='); READ(c);
    IF (a=b)OR(a=c)OR(b=c) THEN WRITELN('YES')
                ELSE WRITELN('NO')
END.
```

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Բաղադրյալ պայմանը ծրագրավորելիս այն կազմող յուրաքանչյուր պարզ պայման պետք է առնել ( ) փակագծերի մեջ:
- Եթե պայմանի օպերատորները ներդրված են, ապա հերթական *ELSE*-ը համապատասխանում է դրան նախորդող մոտակա *IF*-ին:

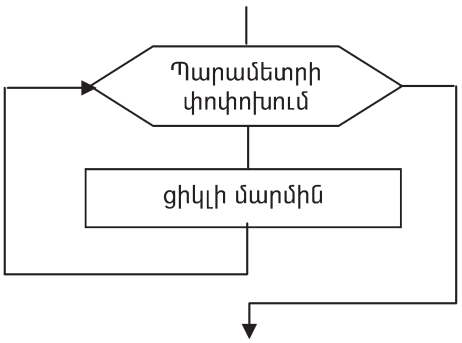
1. **Ո՞ր ալգորիթմն է համարվում ճյուղավորված:**
2. **Ճյուղավորված ալգորիթմների ծրագրավորման նպատակով ի՞նչ օպերատոր է կիրառվում:**
3. **Քանի՞ հնարավոր տեսք ունի պայմանի օպերատորը:**
4. **Բաղադրյալ պայմանը կազմող համախմբելի պարզ պայմանների բլոկները ո՞ր ճյուղերով են կցվում միմյանց և ինչպե՞ս են ծրագրավորվում:**
5. **Արտածել *YES*, եթե տրված թիվը պատկանում է *[-10;0]* կամ *[2;20]* միջակայքերին, հակառակ դեպքում՝ *NO* հաղորդագրությունը:**
6. **Հաշվել և արտածել տրված երեք իրարից տարբեր թվերից մեծի արժեքը:**
7. **Հաշվել և արտածել տրված երեք իրարից տարբեր թվերից փոքրի արժեքը:**
8. **Տրված են երեք դրական թվեր: Արտածել *YES*, եթե այդպիսի երկարություններ ունեցող հատվածներով հնարավոր է եռանկյունի կառուցել, հակառակ դեպքում՝ *NO* հաղորդագրությունը:**
9. **Արտածել *2*, եթե տրված թիվը երկնիշ է, *3*, եթե եռանիշ է, հակառակ դեպքում՝ *0*:**
10. **Տրված է եռանիշ թիվ: Արտածել *1*, եթե եռանիշ թվի թվանշանների գումարը զույգ է, հակառակ դեպքում՝ *0*:**

§2.11. ԿՐԿՆՈՒԹՅԱՆ (ՑԻԿԼԻ) ՕՊԵՐԱՏՈՐՆԵՐ

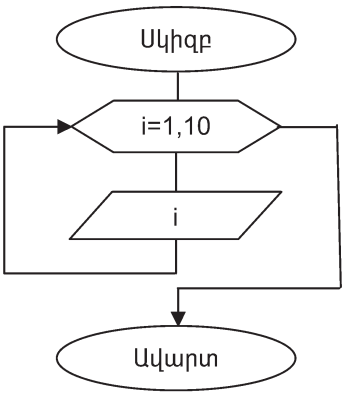
9-րդ դասարանի դասընթացից արդեն ծանոթ եք կրկնվող կամ, այսպես կոչված, ցիկլային բնույթ կրող գործընթացներին: Վերհիշենք, որ ալգորիթմներում գործողությունը կամ գործողությունների խումբը որոշակի անգամ կրկնելու նպատակով կիրառում են պարամետրով ցիկլային կառուցվածքներ, որոնք բլոկ-սխեմաներում ներկայացվում են *մոդիֆիկացիայի բլոկի* միջոցով տրվող ընդհանրական սխեմայով (նկ. 2.9.):

Կրկնվող գործընթացներին ծանոթանալու նպատակով դիտարկենք հետևյալ խնդիրը՝ արտածել *1*-ից *10* միջակայքի ամբողջ թվերը:

Կազմենք խնդրի լուծման բլոկ-սխեման (նկ. 2.10.):



Նկ. 2.9. Պարամետրով ցիկլային կառուցվածքի ընդհանրական սխեմա



Նկ. 2.10. 1-ից 10 միջակայքի ամբողջ թվերի արտածման բլոկ-սխեմա

Այստեղ ցիկլում ներառված միակ՝ *i* փոփոխականի արժեքն արտածելու գործողության կրկնությունների քանակը հայտնի է՝ *10*. այսպիսի գործառույթները ծրագրավորելու համար *պարամետրով ցիկլի օպերատոր* են կիրառում, որը *Պասկալում* ունի հետևյալ ընդհանուր տեսքը.

FOR ցիկլի պարամետր := ցիկլի պարամետրի առաջին արժեք TO ցիկլի պարամետրի վերջնական արժեք DO ցիկլի մարմին:

Ցիկլի պարամետրն օգտագործվում է ցիկլի պարամետրի արժեքը սկզբնականից մինչև վերջնականը *1*-ով ավելացնելու համար: Այս օպերատորն անվանում են *ցիկլի աճող պարամետրով* օպերատոր, որտեղ *FOR, TO* և *DO* բառերն առանցքային բառեր են:

Գրենք նկ. 2.10.-ում բերված բլոկ-սխեմային համապատասխանող ծրագիրը.

```
PROGRAM Par_Cikl1;
VAR i:BYTE;
BEGIN
    FOR i:=1 TO 10 DO                    {1}
        WRITELN(i)                       {2}
END.
```

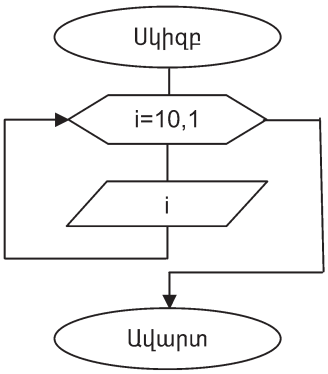
{1} տողում գրվածի համաձայն՝ ծրագրի սկզբում ցիկլի *i* պարամետրը ստանում է նախնական *1* արժեք, որը համեմատվում է պարամետրի վերջնական արժեք հանդիսացող *10*-ի հետ. եթե պարզվեր, որ պարամետրի նախնական *1* արժեքն ավելի մեծ է, քան վերջնական *10* արժեքը՝ ցիկլը (ոչ մի անգամ չիրագործելով ցիկլի մարմնում ներառված հրամանը) կավարտեր աշխատանքը: Քանի որ այդպես չէ, իրականացվում է {2} տողում ներառված հրամանը՝ արտածելով *i*-ի ընթացիկ արժեքը՝ *1* թիվը: Այնուհետև վերադարձ է կատարվում ցիկլի սկիզբ՝ ավտոմատ կերպով (ըստ պարամետրով ցիկլի գործողության սկզբունքի) նախապես *1*-ով մեծացնելով ցիկլի *i* պարամետրի ընթացիկ արժեքը, և ամեն ինչ (պարամետրի ընթացիկ արժեքի համեմատումը վերջնականի հետ և այլն) կրկնվում է նորից: Ցիկլն ավարտում է աշխատանքը, երբ *i* պարամետրի արժեքը դառնում է հնարավոր վերջնականից՝ *10*-ից մեծ:

Եթե անհրաժեշտ լիներ *10*-ից մինչև *1*-ն ընկած թվերն արտածել *10, 9, 8, ..., 1* հաջորդականությամբ, ապա այս գործընթացը (երբ ցիկլի պարամետրը նվազելով է հասնում վերջնական արժեքին), ծրագրավորվում է ցիկլի *նվազող պարամետրով օպերատորի* միջոցով, որի տեսքն այսպիսին է.

FOR ցիկլի պարամետր := ցիկլի պարամետրի առաջին արժեք DOWNTO ցիկլի պարամետրի վերջնական արժեք DO ցիկլի մարմին;

Այստեղ կիրառված *DOWNTO*-ն նույնպես առանցքային բառ է:

Բերենք *10*-ից *1* թվերն արտածելու խնդրի բլոկ-սխեման ու ծրագիրը:



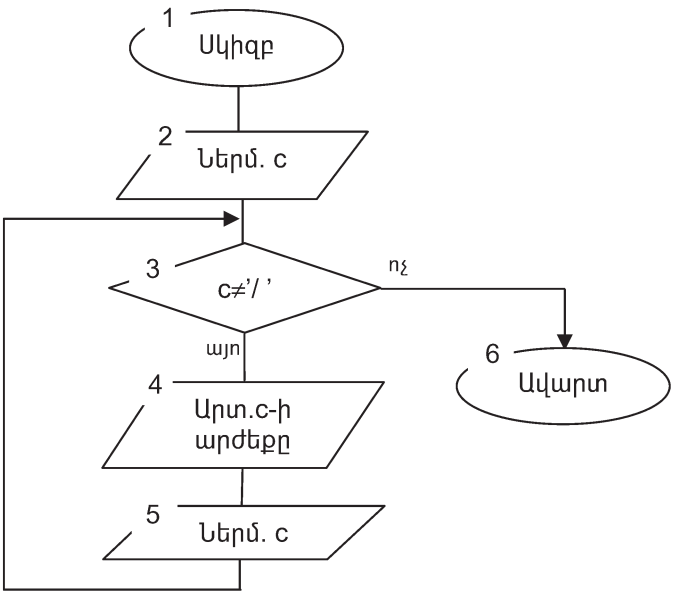
Նկ. 2.11. 10-ից 1 միջակայքի ամբողջ թվերի արտածման բլոկ-սխեմա

```
PROGRAM Par_Cikl2;
VAR i:BYTE;
BEGIN
    FOR i:=10 DOWNTO 1 DO                    {1}
        WRITELN(i)
END.
```

Ծրագրի {1} տողում նախ *i* պարամետրը ստանում է *10* արժեքը. ցիկլի օպերատորը *DOWNTO*-ով գրելու դեպքում ցիկլը աշխատանքը կավարտեր այս փուլում, եթե պարզվեր, որ պարամետրի ընթացիկ արժեքը փոքր է վերջնականից: Քանի որ *10*-ը փոքր չէ *1*-ից՝ իրագործվում է ցիկլի մարմինը՝ արտածելով *i*-ի արժեքը (*10*), որից հետո վերադարձ է կատարվում նորից ցիկլի սկիզբ՝ այս անգամ արդեն պարամետրի ընթացիկ արժեքը *1*-ով ավտոմատ պակասեցնելով: Ամբողջ գործընթացը նորից կրկնվում է սկզբից՝ պարամետրի ընթացիկ *9* արժեքը համեմատվում է վերջնականի՝ *1*-ի հետ և այլն:

Այժմ կրկնողական բնույթի մեկ այլ գործընթաց ուսումնասիրենք: Ենթադրենք՝ անհրաժեշտ է արտածել ստեղնաշարից ներմուծված պայմանանշանը, քանի դեռ ներմուծվածը ‘/ ‘ պայմանանշանը չէ:

Կազմենք այս գործընթացն ապահովող ծրագրի բլոկ-սխեման.



Նկ. 2.12. Նախապայմանով ցիկլային ալգորիթմի օրինակ

Այս բլոկ-սխեմայով նույնպես կրկնողական գործընթաց է իրականացվել, սակայն, ի տարբերություն քննարկված նախորդ դեպքերի, այստեղ գործողությունների կրկնության քանակն անհայտ է: Նման գործընթացներ ծրագրավորելու համար *Պասկալում* նախատեսված են ցիկլի երկու՝ *նախապայմանով* ու *վերջնապայմանով* օպերատորներ:

Դիտարկենք *ցիկլի նախապայմանով օպերատորը*, որի ընդհանուր տեսքը հետևյալն է.

WHILE a DO ցիկլի մարմին;

WHILE-ը և *DO*-ն առանցքային բառեր են, *a*-ն տրամաբանական արտահայտություն է, ցիկլի մարմինը՝ օպերատոր կամ բլոկ:

Նախապայմանով ցիկլի օպերատորի մարմինը կրկնվելով իրագործվում է այնքան, քանի դեռ ցիկլի կրկնության պայմանի՝ *a* տրամաբանական արտահայտության արժեքը ճշմարիտ (*TRUE*) է, և ավարտվում է այն կեղծ (*FALSE*) դառնալու դեպքում:

Այսպիսով, նախապայմանով ցիկլի մարմինը կարող է ոչ մի անգամ չիրագործվել՝ եթե *a*-ն ի սկզբանե ունենա *FALSE* արժեք, իսկ մյուս դեպքում էլ անվերջ կրկնվել, եթե *a*-ն երբեք *FALSE* արժեք չստանա:

Գրենք նկ. 2.11.-ում բերված բլոկ-սխեմային համապատասխանող ծրագիրը.

```
PROGRAM Nax_Cikl;
VAR c:CHAR;
BEGIN WRITE('Որևէ ստեղն սեղմեք');
        READLN(c);
        WHILE c< >'/ ' DO                                    {1}
  BEGIN WRITELN('Ներմուծվել է ' , ' ' : 3, c, ' ' : 2, 'պայմանանշանը');
              READLN(c)
  END
END.
```

Այսպիսով, {1} տողում ներառված նախապայմանով ցիկլի օպերատորը նախ որոշում է *C<>'/ '* տրամաբանական արտահայտության արժեքը, և եթե այն *TRUE* է (ներմուծվածը '/ ' պայմանանշանը չէ), ապա իրականացվում են *BEGIN* և *END*-ի մեջ ներառված օպերատորները. այստեղ ցիկլի մարմինը բլոկ է կազմում, քանի որ կրկնվողը մեկ օպերատոր չէ, այլ մեկից ավելի:

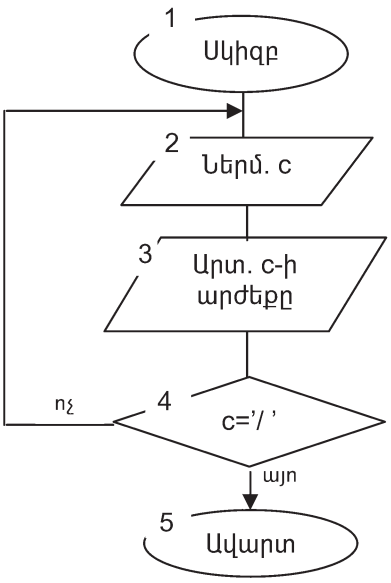
Այժմ ծանոթանանք *վերջնապայմանով* կամ, այսպես կոչված, *հետպայմանով ցիկլի օպերատորին*: Սրա ընդհանուր տեսքը հետևյալն է.

```
REPEAT
   ցիկլի մարմին
UNTIL տրամաբանական արտահայտություն;
```

որտեղ *REPEAT*-ը և *UNTIL*-ը առանցքային բառեր են:

REPEAT և *UNTIL* բառերի միջև ներառվող ցիկլի մարմինը կազմվում է մեկ կամ մի քանի օպերատորներով, որոնք կրկնվելով իրագործվում են այնքան, քանի դեռ *UNTIL*-ի տրամաբանական արտահայտությունն ունի *FALSE* արժեք: Այս օպերատորի առանձնահատկությունն այն է, որ ցիկլի մարմինը, անկախ *UNTIL*-ի տրամաբանական արտահայտության արժեքից, գոնե մեկ անգամ իրագործվում է:

Վերը քննարկված խնդրի բլոկ-սխեման այժմ կառուցենք հետպայմանով, ցիկլի օպերատորի միջոցով իրագործելու համար.



Նկ. 2.13. Հետպայմանով ցիկլային ալգորիթմի օրինակ

Գրենք համապատասխան ծրագիրը.

```
PROGRAM Het_Cikl;
VAR c:CHAR;
BEGIN
      REPEAT
          READLN(c);
          WRITELN('Ներմուծված պայմանանշանը' , ' ՝ ' : 3, c, ' – ն է')
      UNTIL c='/ '
END.
```

Ի տարբերություն նույն խնդրի նախապայմանով ցիկլի օգնությամբ լուծված տարբերակի, այստեղ ցիկլի ավարտից առաջ ներմուծված '/' պայմանանշանը ևս կարտածվի էկրանին:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Ցիկլի ցանկացած օպերատորի աշխատանք կարելի է վաղաժամկետ ավարտել՝ դրա մարմնում *BREAK* օպերատոր կիրառելով:
- Պարամետրով ցիկլի պարամետրը ցանկացած կարգային տիպի փոփոխական է:
- Եթե ցիկլի պարամետրի փոփոխման քայլը *1* կամ *-1* չէ, կամ, եթե այն կարգային տիպի չէ (օրինակ՝ իրական թիվ է), ապա պարամետրով ցիկլի փոխարեն պետք է կիրառել նախապայմանով կամ հետպայմանով ցիկլի օպերատորներից որևէ մեկը:
- Հետպայմանով ցիկլի մարմինը, եթե նույնիսկ մեկից ավելի օպերատորներ է ներառում, ապա կարիք չկա առնելու *BEGIN* և *END* բառերի միջև, քանի որ *REPEAT*-ն այս դեպքում հանդիսանում է նաև ցիկլի մարմնի սկիզբ, իսկ *UNTIL*-ը՝ վերջ:

1. **Բերեք ցիկլային բնույթ կրող որևէ գործընթացի օրինակ:**
2. **Պարամետրով ցիկլի քանի՞ օպերատոր գիտեք:**
3. **Հնարավո՞ր է արդյոք, որ նախապայմանով ցիկլի մարմինը ոչ մի անգամ չիրագործվի. եթե՝ այո, ապա ո՞ր դեպքում:**
4. **Հետպայմանով ցիկլի մարմինն ամենաքիչը քանի՞ անգամ է կատարվում:**
5. **Որոշել [*1;100*] միջակայքի *3*-ին բազմապատիկ տարրերի գումարը.**
   ա) աճող պարամետրով ցիկլի կիրառմամբ,
   բ) նվազող պարամետրով ցիկլի կիրառմամբ,
   գ) նախապայմանով ցիկլի միջոցով,
   դ) հետպայմանով ցիկլի միջոցով:
6. **Հաշվել և արտածել այն երկնիշ թվերի գումարը, որոնք բազմապատիկ են *3*-ին:**

§2.12. ՄԻԱՉԱՓ ԶԱՆԳՎԱԾՆԵՐ

Ծրագրավորման մեջ, բացի պարզ տիպերից, կիրառում են նաև, այսպես կոչված, *կառուցվածքային տիպեր*: Սրանք բնորոշվում են տվյալ տիպը կազմող տարրերի բազմությամբ, այսինքն՝ կառուցվածքային տիպի փոփոխականը կամ հաստատունը միշտ մի քանի տարրերից է բաղկացած:

Կառուցվածքային տիպերից կուսումնասիրենք *զանգվածները*: Զանգվածի բաղադրիչները միևնույն տիպի են: Յուրաքանչյուր բաղադրիչ ունի հերթական համար՝ *ինդեքս*, որի միջոցով կարելի է դիմել այդ տարրին: Եթե զանգվածի տարրին դիմելիս միայն մեկ ինդեքս է կիրառվում, ապա այդ զանգվածն անվանում են *միաչափ*: Միաչափ զանգվածները հայտարարում են հետևյալ կերպ.

VAR իդենտիֆիկատորներ : ARRAY[ինդեքսի սկզբին եզր . . ինդեքսի վերին եզր] OF տիպ;

որտեղ *ARRAY, OF* բառերն առանցքային բառեր են, *ինդեքսի սկզբին* և *վերին եզրերը*՝ կարգային տիպի մեծություններ են (սովորաբար կիրառվում է միջակայքային տիպը): Օրինակ՝

```
a:ARRAY[1..10] OF INTEGER;
x:ARRAY[-2..5] OF CHAR;
y:ARRAY[FALSE..TRUE] OF REAL;
```

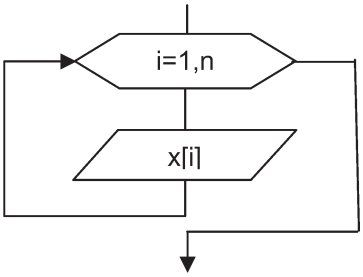
Զանգվածի հայտարարման մեջ *OF*-ից հետո գրված տիպով բնորոշում են զանգվածի տարրերի տիպը: Օրինակ՝ որպես վերը հայտարարված *a* զանգվածի տարրեր կարող են հանդես գալ հետևյալ թվերը՝ *8, -2, 0, 9, -5, 3, 3, 4, 100, -100*, որտեղ *8*-ը զանգվածի առաջին տարրն է կամ, որ նույնն է՝ *a[1]*-ը, *-2*-ը երկրորդ տարրը կամ՝ *a[2]*-ը, ... *-100*-ը զանգվածի *10*-րդ տարրը՝ *a[10]*:

Ինչպես երևում է օրինակից՝ տարրի ինդեքսը գրվում է *[ ]*-երի մեջ:

Զանգվածը կարելի է հայտարարել նաև, օրինակ, հետևյալ կերպ.

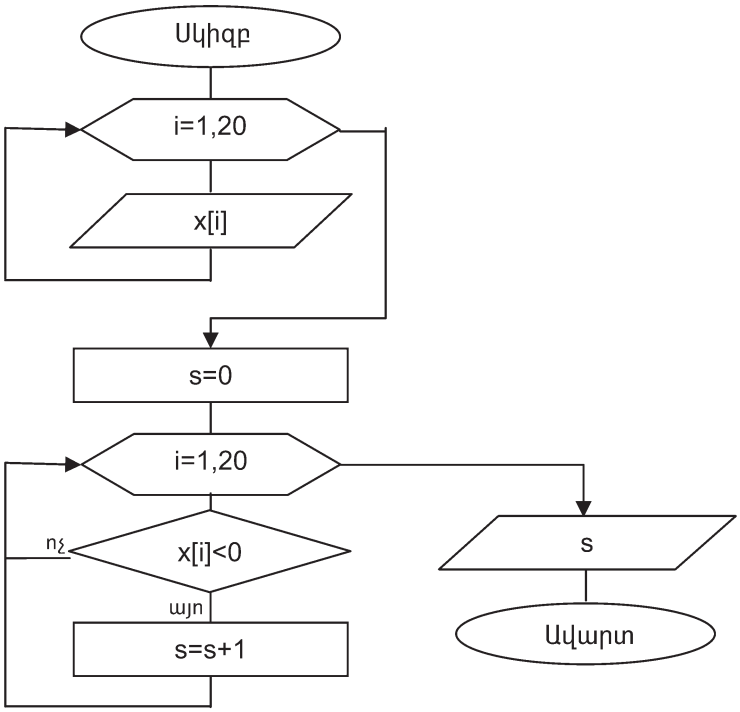
```
TYPE zangvac=ARRAY[1..10] OF REAL;
VAR x:zangvac;
```

Զանգվածների ներմուծումը (արտածումը) կատարվում է ցիկլի միջոցով, որտեղ հերթով ներմուծվում (արտածվում) են *x[1], x[2], ..., x[n]* տարրերի արժեքները:



Նկ. 2.14. Բլոկ-սխեմայում n տարր պարունակող միաչափ զանգվածի տարրերի ներմուծման (արտածման) գործընթացի օրինակ

Դիտարկենք հետևյալ խնդիրը. հաշվել իրական տիպի *20* տարր պարունակող միաչափ զանգվածի բացասական տարրերի քանակը:



Նկ. 2.15. Զանգվածի բացասական տարրերի քանակի հաշվման ալգորիթմ

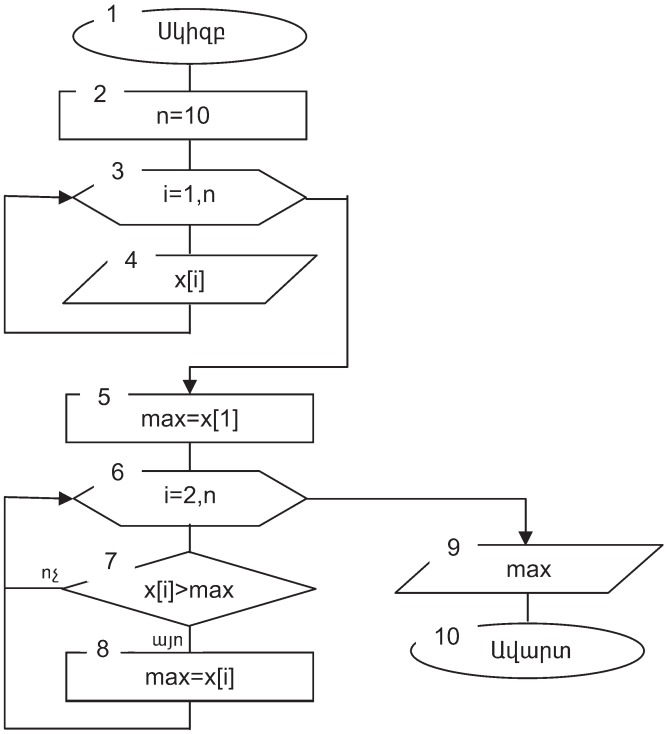
Կազմենք խնդրի լուծման ծրագիրը.

```
PROGRAM Bac_Qanak;
TYPE vektor=ARRAY[1..20] OF REAL;
VAR x:vektor;    {զանգվածի հայտարարումը}
      i,s:BYTE;   {i-ն ինդեքսի և s-ը պահանջվող քանակի համար է}
```

```
BEGIN
      FOR i:=1 TO 20 DO                                          {1}
          BEGIN WRITE('x[',i,']='); READ(x[i]);                  {2}
          END;                                                   {3}
      s:=0;                                                      {4}
      FOR i:=1 TO 20 DO                                          {5}
      IF x[i]<0 THEN s:=s+1;                                     {6}
      WRITELN('զանգ-ի բաց. տարրերի քանակը=',s)                  {7}
END.
```

{1}-ից {3} տողերում *20* իրական թվեր են ներմուծվում, որոնք հաջորդաբար փոխանցվում են *x* զանգվածի տարրերին: Պահանջվող տարրերի քանակը հաշվելու համար *s*-ը {4} տողում սկզբնարժեքավորվում է նախնական *0* արժեքով, այնուհետև {5}-{6}-ում հերթով ստուգվում են զանգվածի բոլոր տարրերը, և, եթե բացասական (<0) տարրեր կան, ապա դրանց քանակը հաշվարկվում է *s:=s+1* հրամանով: {7}-րդ տողում արտածվում է ստացված քանակը:

Հետևյալ խնդրի միջոցով որոշենք իրական տիպի *10* տարր պարունակող միաչափ զանգվածի մեծագույն տարրի արժեքը:



Նկ. 2.16. Զանգվածի մեծագույն արժեքի որոշման ալգորիթմ

Զանգվածի տարրերը ներմուծելուց հետո *5*-րդ բլոկով կատարվել է *max=x[1]* վերագրումը (կարելի է ասել՝ ենթադրվել է, թե մեծագույն տարրը *x[1]*-ն է՝ զանգվածի առաջին տարրը), այնուհետև *6, 7,* և *8*-րդ բլոկներով հաջորդաբար մնացած տարրերը համեմատվել են *max*-ի մեջ պահպանված արժեքի հետ: Եթե առավել մեծ արժեքով *x[i]* տարր է հայտնաբերվել, ապա *8*-րդ բլոկով *max*-ի արժեքը փոխարինվել է այդ առավել մեծ արժեքով: Ցիկլի ավարտին *max*-ի մեջ կլինի փնտրված մեծագույն արժեքը:

Կազմենք ծրագիրը.

```
PROGRAM max;
CONST n=10;
VAR x:ARRAY[1..n] OF REAL;
    i:BYTE; max:REAL;
BEGIN FOR i:=1 TO n DO READ(x[i]);
        max:=x[1];
       FOR i:=2 TO n DO IF x[i]>max THEN max:=x[i];
      WRITELN('max=',max:6:2)
END.
```

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Զանգվածի տարրերը համակարգչի հիշողությունում անընդմեջ հաջորդական տարածք են զբաղեցնում:

1. **Ինչ է զանգվածը:**
2. **Ո՞ր զանգվածներն են անվանում միաչափ:**
3. **Հաշվել և արտածել տրված *n* տարր պարունակող միաչափ զանգվածի զույգ ինդեքս ունեցող տարրերի գումարը:**
4. **Հաշվել և արտածել տրված *n* տարր պարունակող միաչափ զանգվածի տրված *[a; b]* միջակայքին պատկանող տարրերի գումարը:**
5. **Հաշվել և արտածել տրված *n* տարր պարունակող միաչափ զանգվածի այն տարրերի արտադրյալը, որոնք բացարձակ արժեքով փոքր են *t* թվից:**
6. **Հաշվել և արտածել տրված *n* ամբողջ տիպի տարր պարունակող միաչափ զանգվածի զույգ արժեք ունեցող տարրերի քանակը:**
7. **Հաշվել և արտածել տրված *n* տարր պարունակող միաչափ զանգվածի *m* բնական թվին բազմապատիկ տարրերի արտադրյալը:**

§2.13. ԵՐԿՉԱՓ ԶԱՆԳՎԱԾՆԵՐ

Երկչափ զանգվածի յուրաքանչյուր տարրին դիմելու համար անհրաժեշտ է երկու ինդեքս կիրառել: Երկչափ զանգվածը պատկերացնելու համար դիտենք դասարանի նստարանների շարքերը: Ենթադրենք, դրանք դասավորված են *4* շարքով, իսկ յուրաքանչյուր շարքում *3*-ական սեղաններ կան: Համարակալենք սեղաններն այնպես, որպեսզի այդ համարներով միարժեքորեն որոշվի սեղանի գտնվելու շարքը և շարքում դրա դիրքը:

S[1,1]	*S[1,2]*	*S[1,3]*	*S[1,4]*
S[2,1]	*S[2,2]*	*S[2,3]*	*S[2,4]*
S[3,1]	*S[3,2]*	*S[3,3]*	*S[3,4]*

Ինչպես տեսնում եք, շարքը բնորոշող համարը քառակուսի փակագծերի մեջ առնված թվերից երկրորդն է, իսկ առաջինը՝ տվյալ շարքում նստարանի հերթական համարը: Նման եղանակով կարգավորված տվյալների համախումբն անվանում են *երկչափ զանգված*: Երկչափ զանգվածի տարրի *առաջին ինդեքսը* համարում են տարրի գտնվելու *տողի*, իսկ երկրորդը՝ *սյան* համարը: Նույն սկզբունքով կարելի է սահմանել նաև *բազմաչափ զանգվածները:*

Երկչափ զանգվածներ հայտարարելիս պետք է նշել դրա երկու ինդեքսների փոփոխման միջակայքերը, օրինակ.

VAR x:ARRAY[1..10,1..20] OF REAL;

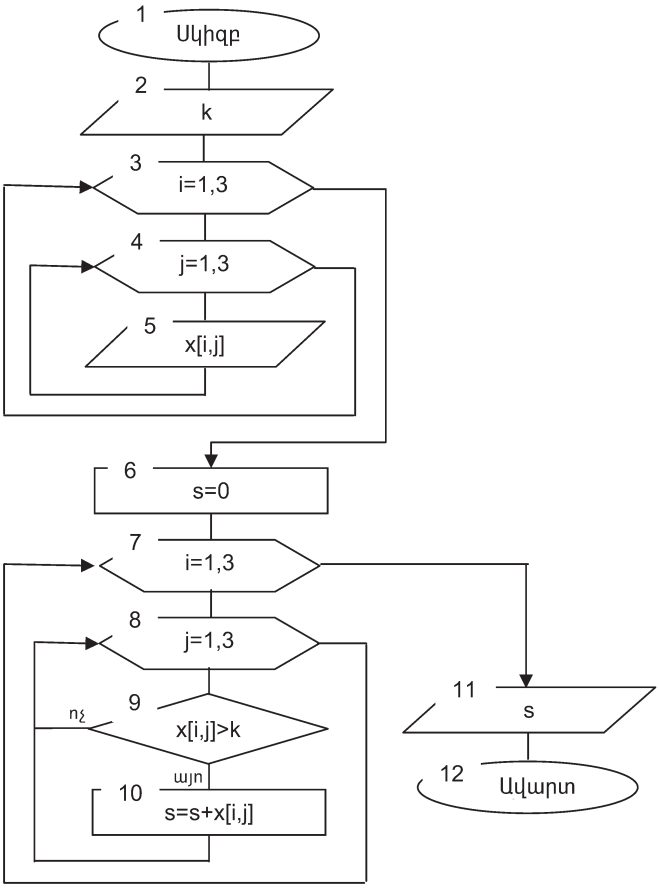
որտեղ փակագծերում *1..10*-ը ցույց է տալիս առաջին ինդեքսի, իսկ *1..20*-ը՝ երկրորդ ինդեքսի փոփոխման տիրույթը:

Երկչափ զանգվածների հետ կապված աշխատանքին առավել մոտիկից ծանոթանալու նպատակով մի քանի խնդիր լուծենք:

Խնդիր. տրված է *3 x 3* (*3* տող և *3* սյուն) ամբողջ տիպի տարրեր պարունակող երկչափ զանգված: Հաշվել զանգվածի տրված *k* ամբողջ թվից մեծ արժեք ունեցող տարրերի գումարը:

Ինչպես երևում է բլոկ-սխեմայից (նկ. 2.17.), երկչափ զանգված ներմուծելու համար մեկը {4} մյուսի {3} մեջ ներդրված ցիկլեր են կիրառվել: *Ներդրված ցիկլերն* աշխատում են հետևյալ կերպ. նախ արտաքին {3} ցիկլի *i* պարամետրը ստանում է իր սկզբնական՝ 1 արժեքը, և ղեկավարումը տրվում է ներդրված {4} ցիկլին, վերջինս ցիկլի սովորական, մեզ արդեն հայտնի սխեմայով է աշխատում, այսինքն՝ *i = 1* արժեքի դեպքում *j*-ն փոփոխվելով *1*-ից *3*՝ ներմուծվում են *i*-րդ (այս պահին՝ *1*-ին) տողի տարրերը, այնուհետև ղեկավարումը կրկին տրվում է {3} բլոկին՝ որտեղ *i*-ն աճելով ստանում է *2* արժեքը, և ամեն ինչ ընթանում է այնպես, ինչպես *i = 1* արժեքի դեպքում, այսինքն՝ այժմ ներմուծվում են *2*-րդ տողի տարրերը: Նույնը կատարվում է

նաև *i = 3*-ի դեպքում: Աշխատանքի այսպիսի ընթացքը հատկանշական է ցանկացած ներդրված ցիկլի համար:



Նկ. 2.17. Երկչափ զանգվածում գումարի հաշվման ալգորիթմ

Այժմ կազմենք նկարագրված բլոկ-սխեմային համապատասխանող ծրագիրը.

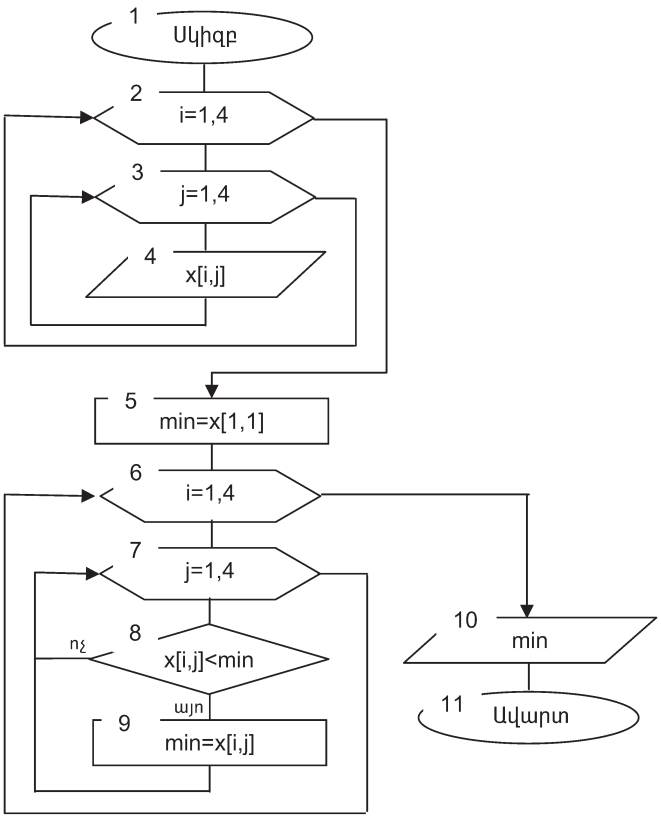
```
PROGRAM Matrici_Khndir;
CONST n=3;
TYPE matric=ARRAY[1..n,1..n] OF INTEGER;
VAR i,j:BYTE; k,s:INTEGER; x:matric;
BEGIN WRITE('k=');READ(k);
FOR i:=1 TO n DO                                    {1}
    FOR j:=1 TO n DO                                {2}
        READ(x[i,j]);                               {3}
s:=0;
FOR i:=1 TO n DO
```

```
        FOR j:=1 TO n DO
            IF x[i,j]>k THEN s:=s+x[i,j];
            WRITELN('k-ից մեծ տարրերի գումարը=',s:10)
    END.
```

Ծրագրում {1} և {2} մեկնաբանությամբ ցիկլերի միջև *BEGIN*-ի անհրաժեշտություն չկա, քանի որ {1}-ի ցիկլում մեկ օպերատոր կա՝ {2}-ը, իսկ {3}-ը {2}-ի միակ կրկնվող օպերատորն է:

Կազմենք հետևյալ խնդրի լուծման բլոկ-սխեման ու ծրագիրը ևս. տրված է *4x4* իրական տիպի տարրեր պարունակող երկչափ զանգված: Հաշվել և արտածել զանգվածի փոքրագույն տարրի արժեքը:

Նախ կազմենք խնդրի լուծման բլոկ-սխեման:



Նկ. 2.18. Երկչափ զանգվածի փոքրագույն տարրի որոշման ալգորիթմ

Նախ *min* փոփոխականի մեջ պահվել է զանգվածի տարրերից առաջինի արժեքը (ընդ որում, կարևոր չէ, թե որ տարրի արժեքն այս պահին կդիտվի որպես փոքրագույն), որից հետո *6*-րդ և *7*-րդ բլոկներով կազմավորված ներդրված ցիկլերի միջոցով ենթադրյալ փոքրագույնի (*min*) հետ համեմատվել են զանգվածի մնացած բոլոր տարրերը և արժեքով առավել

փոքր տարրը *9*-րդ բլոկում վերագրվել է *min*-ին: Վերջում *min*-ի մեջ կմնա զանգվածի փոքրագույն տարրը, որի արժեքն արտածվել է *10*-րդ բլոկով:

Կազմենք ծրագիրը.

```
PROGRAM Min_Matric;
CONST n=4;
VAR x:ARRAY[1..n,1..n] OF REAL;
    i,j:BYTE; min:REAL;
BEGIN  FOR i:=1 TO n DO
          FOR j:=1 TO n DO
             READ(x[i,j]);
    min:=x[1,1];
    FOR i:=1 TO n DO
       FOR j:=1 TO n DO
          IF x[i,j]<min THEN min:=x[i,j];
    WRITE('min=',min:10:2)
END.
```

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Զանգվածը, անկախ իր չափողականությունից, չի կարող *65520* բայթից ավելի հիշողություն զբաղեցնել:

1. Կարող են երկչափ զանգվածի առաջին տողի տարրերը լինել սիմվոլային տիպի, իսկ մնացած տարրերը՝ օրինակ՝ ամբողջ տիպի:
2. Եթե զանգվածի տարրը բնորոշվում է որպես *x[a,b,c]*, ապա զանգվածը.
   ա) երկչափ է, բ) միաչափ է, գ) եռաչափ է, դ) *a*b*c* չափի է:
3. Կարող է արդյոք զանգվածի ինդեքսը լինել իրական թիվ:
4. Հաշվել և արտածել *2x3* տարր պարունակող երկչափ զանգվածի տարրերի արտադրյալը:
5. Հաշվել և արտածել *3x3* տարր պարունակող երկչափ զանգվածի դրական տարրերի գումարը:
6. Հաշվել և արտածել *3x3* տարր պարունակող երկչափ զանգվածի կենտ արժեք ունեցող տարրերի արտադրյալը:
7. Հաշվել և արտածել *5x5* տարր պարունակող երկչափ զանգվածի *5*-ին բազմապատիկ տարրերի քանակը:

3. ՀԵՌԱՀԱՂՈՐԴԱԿՑՄԱՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐ

§3.1. *HTML*-ՀԻՊԵՐՏԵՔՍՏԻ ՆՇԱԳՐՄԱՆ ԼԵԶՈՒ

11-րդ դասարանում *WEB*-էջեր ստեղծելու որոշ փորձառություն ձեռք բերեցիք: Դասընթացի այս բաժնում կծանոթանաք այդ նպատակին ուղղված մի շարք նոր հնարավորությունների:

Նախ համառոտ կրկնենք ձեզ հայտնի նյութը:

Հիշեցնենք, որ *HTML*- փաստաթուղթը *սկսվում է <html>* թէգով և *ավարտվում </html>* թէգով: Փաստաթուղթը բաղկացած է երկու հիմնական մասից՝ *ծանուցման բաժին* ու *փաստաթղթի մարմին*: Ծանուցման բաժինը ներառվում է *<head>* և *</head>* թէգերի մեջ: Այս բաժինը ներառում է էջի վերնագիրը և տեխնիկական այլ բնութագրեր: Էջի վերնագիրը տրվում է *<title> ... </title>* *մեկնարկի* և *ավարտի* թէգերի միջև. սա ծանուցման բաժնի միակ պարտադիր թէգն է:

HTML-փաստաթղթի մարմինը ներառվում է *<body>* և *</body>* զույգ թէգերի մեջ: Օրինակ՝

```
<html>
    <head><title> Փաստաթղթի նկարագրությունը </title></head>
    <body>
        Փաստաթղթի տեքստը
    </body>
</html>
```

Ինչպես հիշում եք, *HTML* լեզվում վերնագրերի *6* մակարդակներ են նախատեսված՝ *h1, h2, ..., h6*, որոնք տրվում են համապատասխան զույգ թէգերի օգնությամբ, ընդ որում, *<h1>*-ը առաջին մակարդակի վերնագիր է և ունի ամենամեծ տառաչափը, իսկ *<h6>*-ը՝ *6*-րդ մակարդակի, որի տառաչափն ամենափոքրն է:

Ստեղծվող փաստաթղթում վերնագրի հավասարեցման ձևը կարելի է սահմանել *align* հատկանիշի միջոցով՝ հետևյալ կերպ. *right* – ըստ աջ եզրի, *left* – ըստ ձախ եզրի, *center* – ըստ կենտրոնի:

Օրինակ՝ *<h1 align="center">վերնագիր</h1>:*

HTML-փաստաթղթում պարբերությունները տրվում են զույգ՝ * * և * * թէգերի օգնությամբ. օրինակ՝

```
<p>առաջին պարբերություն</p><p>երկրորդ պարբերություն</p>
```

Պարբերության հավասարեցումը նույնպես իրականացվում է align հատկանիշի օգնությամբ, որն այստեղ կարող է ընդունել հետևյալ արժեքներից որևէ մեկը. *left* – ըստ ձախ եզրի, *right* – ըստ աջ եզրի, *center* – ըստ կենտրոնի, *justify* – ըստ աջ և ձախ եզրերի: Օրինակ՝ *<p align="justify">*

Տեքստում տողադարձն իրականացվում է *
* առանձին թէգով:

Պարբերություններն իրարից առանձնացնելու համար կարելի է նաև դրանց միջև հորիզոնական գիծ տանել՝ օգտվելով *<hr />* թէգից: Դրվող գծի դիրքը, երկարությունն ու հաստությունը տրվում են թէգի հատկանիշներում, օրինակ՝

<hr / align="center" size="2px" width="20%">

Հիշեցնենք, որ *HTML*-ում *համարակալված տարրերով ցուցակներ* ստեղծելու համար հատուկ թեգեր են կիրառվում: Նման ցուցակը պետք է սկսվի *
* և ավարտվի *
* թէգերով, իսկ ցուցակի յուրաքանչյուր տարր պետք է ներառվի *<li>* և *</ li>* թէգերի միջև:

<ol> թեգում *type* հատկանիշով կարելի է նշել ցուցակի տարրերը համարակալելու եղանակը.

- *A – լատինական մեծատառերով՝ A,B,C...,*
- *a – լատինական փոքրատառերով՝ a,b,c...,*
- *I – հռոմեական I,II,III... թվերով,*
- *i – հռոմեական i,ii,iii... թվերով,*
- *1 – արաբական թվերով՝ 1,2,3...,*

Օրինակ՝ *<ol type="a">*: Հնարավորություն կա նշելու նաև համարակալման ելակետային արժեքը, որի համար կիրառվում է *start* հատկանիշը: Օրինակ՝ ըստ *<ol type="1" start="5">* :

Չհամարակալված ցուցակ ստեղծելու համար կիրառվում են *
* և *
* թէգերը: Այստեղ ևս ցուցակի յուրաքանչյուր տարր պետք է ներառված լինի *<li>* և *</ li>* թէգերի միջև: *<ul>* թէգում *type* հատկանիշով որոշվում է նշիչի արտաքին տեսքը: Այն կարող է ընդունել հետևյալ արժեքներից որևէ մեկը. *circle* – շրջանագիծ, *square* – քառակուսի: Օրինակ՝ *<ul type="circle">* :

Հիշեցնենք, որ հիպերտեքստային հղում կարելի է ստեղծել *<a>* զույգ թէգի օգնությամբ: Այն միայն մեկ պարտադիր հատկանիշ ունի՝ *href*-ը, որի արժեքն այն փաստաթղթի հասցեն ու անվանումն է, որին ուղղված է հղումը: Օրինակ՝

նախարարություն :

Էջում պատկեր տեղադրելու համար նախատեսված է *<img>* թէգը: *Մեկնարկի <img>* թէգը պետք է ներառի *src* հատկանիշը, որի արժեքը տեղադրվող պատկերի անվանումն ու հասցեն է: Օրինակ՝ *<img src="c:\picture.jpg">* :

Պատկերն այլ, *անհրաժեշտ չափերով տեղադրելու համար <img>* թէգում նախատեսված են *width* և *height* հատկանիշները, որոնցով սահմանվում են

տեղադրվող պատկերի համապատասխան լայնությունն ու բարձրությունը: Օրինակ՝

<img src="c:\patker.jpg" width="90px" height="80px"> :

Տեքստում պատկերի դիրքը կարելի է սահմանել *align* հատկանիշի օգնությամբ, որը կարող է ընդունել հետևյալ արժեքներից որևէ մեկը.

- *bottom* – պատկերի ստորին եզրը հավասարեցվում է տեքստային տողի հիմքին,
- *middle* – տողն անցնում է պատկերի կենտրոնով,
- *top* – պատկերի վերին եզրը հավասարեցվում է տեքստային տողի հիմքին,
- *left* – պատկերը տեղադրվում է էջի ձախ մասում, իսկ հաջորդող տեքստը՝ դրանից աջ,
- *right* – պատկերը տեղադրվում է էջի աջ մասում, իսկ հաջորդող տեքստը՝ դրանից ձախ:

Օրինակ՝ *<img spr="c:\patker.jpg" align="top"> :*

Ֆոնային պատկեր տեղադրելու համար կարելի է օգտվել *body* թէգի *background* հատկանիշից, որի արժեքը պատկերի անվանումն ու հասցեն է: Օրինակ՝

<body background="c:\patker1.gif"> :

Հիշեցնենք, որ *HTML*-ում *աղյուսակ կարելի է սփեղծել <table>* զույգ թէգի օգնությամբ: Աղյուսակում վերնագիրը տրվում է *<caption>*, իսկ տողերը՝ *<tr>* զույգ թէգերի օգնությամբ:

Աղյուսակի յուրաքանչյուր տող *բջիջներ* է պարունակում: Վերնագրում բջիջները տրվում են *<th>*, իսկ տողերում՝ *<td>* զույգ թէգերի կիրառմամբ:

Էկրանին աղյուսակի հորիզոնական դիրքը տալու համար նախատեսված է align հատկանիշը, որը կարող է ընդունել հետևյալ արժեքներից որևէ մեկը. *left* – ձախ եզրում, *right* – աջ եզրում, *center* – կենտրոնում: Օրինակ՝ *<table align="center">* :

<td> և *<th>* թէգերի դեպքում էլ *align* հատկանիշով կարելի է փոխել արտածվող տվյալների հորիզոնական հավասարեցման ձևը՝ դրան վերագրելով հետևյալ արժեքներից որևէ մեկը. *left* – ըստ ձախ եզրի, *right* – ըստ աջ եզրի, *center* – ըստ կենտրոնի:

valign հատկանիշի օգնությամբ կարելի է աղյուսակի բջիջներում տվյալների հավասարեցումն իրականացնել ուղղաձիգ ուղղությամբ: Այս հատկանիշը կարող է ընդունել հետևյալ հնարավոր արժեքները. *top* – ըստ բջջի վերին եզրի, *bottom* – ըստ բջջի ստորին եզրի, *middle* – ըստ բջջի կենտրոնի:

Աղյուսակի լայնության ու բարձրության չափերը կարելի է սահմանել *width* և *height* հատկանիշների միջոցով: Օրինակ՝

<table width="20px" height="10px"> :

§3.2. *CSS* – ՈՃԵՐԻ ԿԱՍԿԱԴԱՅԻՆ ԱՂՅՈՒՍԱԿՆԵՐ

CSS–ը (*Cascading Style Sheets*) *WEB*–էջերի դիզայնը կարգավորելու տեխնոլոգիա ընձեռող լեզու է, որն էականորեն հարստացնում է էջի արտաքին տեսքը ձևավորելու հնարավորություններն ու հեշտացնում դրա խմբագրման գործընթացը:

Եթե հիպերտեքստի նշագրման *HTML* լեզուն հիմնականում սահմանում է *WEB*–էջի կառուցվածքը, ապա *CSS*–ը թույլատրում է ոճ սահմանել էջի յուրաքանչյուր օբյեկտի համար և այն պահպանել առանձին ֆայլում: *CSS*–ի օգնությամբ կարելի է կայքի էջերի որոշ պարամետրեր (օրինակ՝ տառաչափը) փոփոխել՝ անփոփոխ թողնելով սերվերում պահվող *HTML* փաստաթուղթը: Անհրաժեշտության դեպքում կիրառողի բրաուզերը կարող է դիմել *CSS*–ով ստեղծված *ոճերի ֆայլին* և էջին անհրաժեշտ տեսք տալ:

CSS–ում սահմանվող ոճն ունի հետևյալ ընդհանուր գրելաձևը.

```
ընտրիչ
        {
          հատկանիշ1 : արժեք1;
          հատկանիշ2 : արժեք2;
          հատկանիշ3 : արժեք3;
        }
```

Ընտրիչը *HTML* լեզվի թէգերի անվանումներով կազմվող գրառում է, որը որոշում է, թե *WEB*–էջի որ թէգերի համար և ինչպես է պետք կիրառել ձևավոր փակագծերում բերված կանոնները:

Ընտրիչին հաջորդող {...} ձևավոր փակագծերում ընդգրկվում են ոճը ներկայացնող կանոնները՝ բաժանված կետ-ստորակետերով (;): Կանոնները տրվում են *հատկանիշ:արժեք;* գրառմամբ, որտեղ *հատկանիշը* *WEB*–էջի դիզայնի որևէ բաղադրիչ է, արժեքը՝ տվյալ բաղադրիչի հնարավոր արժեքներից որևէ մեկը:

CSS–ում ընտրիչ գրելու տարբեր ձևեր կան: Ծանոթանանք մի քանիսին.

- եթե ընտրիչում տրվի թէգի անվանումը, ապա սահմանված ոճը կկիրառվի *WEB*–էջում առկա բոլոր նման թէգերում: Օրինակ՝ *a {...}* արտահայտությամբ կազմված ոճը կկիրառվի տվյալ *WEB*–էջում ընդգրկված բոլոր հղումների դեպքում,
- եթե ընտրիչում իրարից ստորակետերով բաժանված մի քանի թէգերի անվանումներ գրվեն, ապա սահմանված ոճը կկիրառվի թվարկված բոլոր թէգերում: Օրինակ՝ *h1,h2,p {...}* ոճը կկիրառվի *WEB*–էջում առկա *h1* և *h2* մակարդակի բոլոր վերնագրերի ու պարբերությունների համար,
- եթե ընտրիչում իրարից բացատանիշերով բաժանված մի քանի թէգերի անվանումներ գրվեն, ապա սահմանված ոճը կկիրառվի թվարկ-

ված հաջորդականությամբ ներդրվող վերջին բաղկացուցիչ թէգի համար: Օրինակ՝ *p a {...}* ոճը կկիրառվի պարբերություններում նե-րառված բոլոր հղումների վրա:

- եթե ընտրիչը ներկայացվի միայն * պայմանանշանով, ապա սահման-ված ոճը կկիրառվի *WEB*-փաստաթղթի բոլոր տարրերի վրա: Օրինակ՝ **{...}* ոճը կկիրառվի *WEB*-էջի բոլոր թէգերի վրա:

CSS-ում կիրառվող շատ հրահանգներ նման են *HTML*-ի համապատաս-խան հրահանգներին: Օրինակ, *WEB*-էջի ֆոնի գույնը *HTML*-ում կարելի է սահմանել *<body bgcolor="#66CDAA">*, իսկ *CSS*-ում՝ *body {background-color: #66CDAA;}* արտահայտությամբ:

Ոճ սահմանող ինֆորմացիան կարող է պահպանվել ինչպես առանձին ֆայլում, այնպես էլ *WEB*-էջի անմիջական կոդում: *CSS*-ով ստեղծված ոճերի նկարագրություններն առանձին ֆայլում իմաստ ունի պահպանել այն դեպ-քում, եթե դրանք պետք է կիրառվեն մի քանի *WEB*-էջերի համար: Այդ նպա-տակով անհրաժեշտ է որևէ տեքստային խմբագրիչի միջավայրում *CSS*-ի հրահանգներով նկարագրել անհրաժեշտ ոճերն ու ստեղծված ֆայլը պահ-պանել *WEB*-սերվերի վրա, իսկ այդ ոճերը կիրառող *WEB*-էջերի կոդերում դիմել (հղում անել) այդ ֆայլին:

Այժմ *HTML*-ում *CSS* ոճերը կիրառելու երեք մեթոդներ քննարկենք:

Մեթոդ 1 (In-line – ներկառուցված). ոճերի աղյուսակն այս դեպքում նկա-րագրվում է *HTML* փաստաթղթի որևէ առանձին թէգի մարմնում՝ *CSS*-ի *style* հատկանիշի օգնությամբ: Այս դեպքում ոճերի աղյուսակի կանոնների ազ-դեցության տիրույթը տվյալ թէգն է: Օրինակ՝ հետևյալ *HTML* ծրագրային կոդում *WEB*-էջի ֆոնի գույնը սահմանվում է կարմիր.

```
<html>
    <head>
      <title>Example</title>
    </head>
      <body style="background-color:red;">
      <p>This is a red page</p>
    </body>
  </html>
```

Մեթոդ 2 (ներքին) – ոճերի աղյուսակը նկարագրվում է *HTML* փաստաթղ-թում, *<style>* և *</style>* թէգերի միջև, որոնք իրենց հերթին գտնվում են փաս-տաթղթի *< head >* և *</head >* թէգերի միջև: Ոճերի աղյուսակի սահմանվող կանոնների ազդեցության տիրույթն այժմ ամբողջ փաստաթուղթն է: Այս դեպքում *WEB*-էջի ֆոնի կարմիր գույնը կսահմանվի հետևյալ կերպ.

```
<html>
    <head>
      <title>Example</title>
      <style type="text/css">
        body {background-color: red;}
      </style>
    </head>
    <body>
      <p>This is a red page</p>
    </body>
  </html>
```

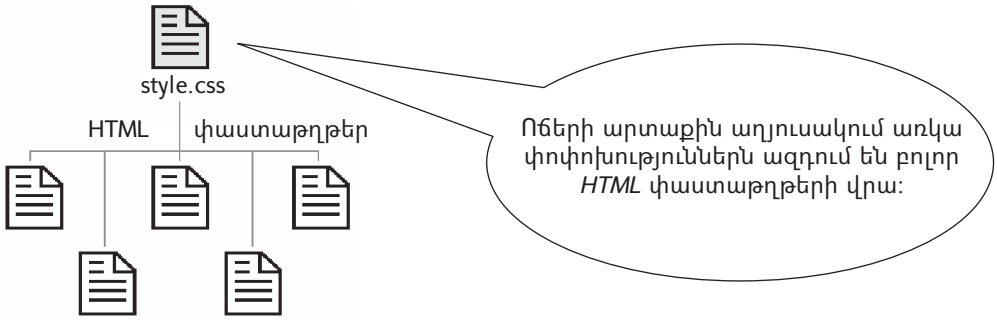
Մեթոդ 3 (արտաքին) – ոճերի աղյուսակը նկարագրվում է *css* ընդլայնմամբ առանձին ֆայլում: *HTML* փաստաթղթից տվյալ ֆայլի հղումն իրականացվում է *<link>* թէգի օգնությամբ, որը տեղակայվում է *header* բաժնում, այսինքն՝ *<head>* և *</head>* թէգերի միջև.

```
<link rel="stylesheet" type="text/css" href="url">
```

Այս թէգի առաջին երկու պարամետրերը պահեստավորված անվանումներ են, որոնք բրաուզերին հայտնում են, որ այս էջում *CSS*-ի ֆայլ է օգտագործվելու: Երրորդ՝ *href* պարամետրը հղում է կատարում ոճերի աղյուսակը պարունակող ֆայլին: Օրինակ՝ ոճերի աղյուսակը պահպանող *style.css* ֆայլին կարելի է հղում կատարել *HTML* կոդի հետևյալ տողով.

```
<link rel="stylesheet" type="text/css" href="style/style.css"/>
```

Այս հղումը բրաուզերին հուշում է, որ *WEB*-էջը խմբագրելու համար պետք է օգտագործի *style.css* ֆայլում պահպանված կանոնները: Նման մոտեցումը հնարավորություն է տալիս բավականաչափ ժամանակ տնտեսել, եթե ոչ մեկ, այլ *WEB*-կայքի բազմաթիվ էջերում է անհրաժեշտ փոփոխություններ մտցնել (նկ. 3.1.): Օրինակ, եթե *WEB*-կայքի մի քանի էջերում է անհրաժեշտ ֆոնների գույները փոխել, ապա կարիք չկա համապատասխան բոլոր էջերի *HTML* փաստաթղթերում փոփոխություններ մտցնել. հերիք է փոփոխել միայն ոճերի աղյուսակի կոդը:



Նկ. 3.1. HTML փաստաթղթերի հղումները ոճերի միևնույն աղյուսակին

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *CSS*-ում կարելի է մեկնաբանություններ ավելացնել, դրանք /* և */ պայմանանշանների միջև առնելով:
- Եթե ոճերի աղյուսակը պահպանող ֆայլն այն սերվերի վրա է, որտեղ պահպանվում են նաև տվյալ ֆայլին հղում ունեցող *WEB-փաստաթղթերը*, ապա *<link>* թէգի *href* պարամետրը կարող է պարունակել այդ ֆայլի հարաբերական հասցեն, հակառակ դեպքում՝ լրիվ հասցեն:

1. **Ինչ է *CSS*-ը:**
2. ***CSS*-ը *HTML*-ում կիրառելու ինչ մեթոդներ գիտեք:**
3. **Ոճերի աղյուսակը արտաքին ֆայլում նկարագրելու դեպքում *HTML* փաստաթղթից *CSS* ֆայլին կատարվող հղումը որտեղ է պետք տեղադրել:**

Լաբորատոր աշխատանք 3.1
Էջերում ոճերի տեղակայման միջոցները

CSS-ին առնչվող լաբորատոր աշխատանքները կատարելուց առաջ *My Documents*-ի ձեր դասարանին հատկացված թղթապանակում ենթաթղթապանակ ստեղծեք: Դրա համար բացեք ձեր դասարանին հատկացված թղթապանակը, մկնիկի ցուցիչը տեղադրեք թղթապանակի որևէ ազատ մասում ու սեղմեք աջ սեղմակը: Բացված ենթատեքստային մենյուից ընտրեք *New*, ապա *Folder* հրամանները: Ներմուծեք ստեղծված թղթապանակի *CSS_** անվանումը, որտեղ *-ի փոխարեն ներմուծեք ձեր դասամատյանի համարը:

1. Ընտրեք համակարգիչ պատկերող որևէ նկար ու այն *computer.png* անվանումով պահպանեք ստեղծված թղթապանակում:
2. *Start* գլխավոր մենյուի *Programs* ենթամենյուի *Accessories* ենթամենյուի *Notepad* հրամանով համանուն տեքստային խմբագրիչի միջավայր մտեք:
3. Ներմուծեք հետևյալ կոդը.

```
<html>
<head>
   <title>lab_2_8</title>
   <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
   <table >
<tr>
          <td colspan="2">
        <h1>ՀԱՄԱԿԱՐԳԻՉ </h1>
           <h1>Ուսումնական կենտրոն</h1>
          </td>
</tr>
<tr height="500">
          <td width="200">
        <ol type=1>
              <li><a href="#"> MS Word </a></li>
              <li><a href="#"> MS Excel</a></li>
            </ol>
          </td>
          <td>
              <img src="computer.png" >
              <h2>Microsoft Office</h2>
              <p>Այս էջը նվիրված է  գրասենյակային  Microsoft Office
                    բազմաֆունկցիոնալ փաթեթին:</p>
          </td>
</tr>
<tr>
          <td colspan="2" >
          <h3>Copyright &copy; 2012</h3>
          </td>
</tr>
   </table>
</body>
</html>
```

4. Ներմուծված փաստաթուղթը պահպանելու նպատակով ընտրեք մենյուի տողի *File* ենթամենյուի *Save As* հրամանը:
5. Բացված պատուհանի *Save as type* դաշտում ընտրեք *All files* տարբերակը:
6. Հայերենով գրված տեքստը պահպանելու համար *Encoding* դաշտում ընտրեք *UTF-8* տարբերակը:
7. *Save in* դաշտում ընտրեք *My Documents*-ի ձեր դասարանին հատկացված թղթապանակը:
8. *FIle Name* դաշտում ներմուծեք *Index.html* անվանումը:

9. Ընտրեք *Notepad* տեքստային խմբագրիչի մենյուի տողի *File* ենթամենյուի *New* հրամանը:
10. Ներմուծեք հետևյալ կոդը.

```
table, td{
        border: 1px solid green;
}
```

11. Ներմուծված փաստաթուղթը պահպանելու նպատակով ընտրեք մենյուի տողի *File* ենթամենյուի *Save As* հրամանը:
12. *Save in* դաշտում ընտրեք *My Documents*–ի ձեր դասարանին հատկացված թղթապանակը:
13. *FIle Name* դաշտում ներմուծեք *Style.css* անվանումը:
14. *Start* գլխավոր մենյուի *Internet Explorer* հրամանով կամ  գործիքով *Internet Explorer*–ի միջավայր մտեք:
15. Ընտրեք մենյուի տողի *File* ենթամենյուի *Open* հրամանը:
16. Սեղմեք *Browse* կոճակը, բացված պատուհանի *Look in* դաշտում ընտրեք *Notepad* խմբագրիչի միջավայրում ձեր ստեղծած *Index.html* ֆայլն ու սեղմեք *Open* կոճակը:
17. Ֆայլի հասցեին վերաբերող հարցումը հաստատեք *OK* կոճակով և էկրանին կհայտնվի ոճերի աղյուսակ կիրառող ձեր առաջին *WEB*-էջը.

18. Ավարտեք աշխատանքը՝ տեքստային խմբագրիչն ու բրաուզերը փակելով:

§3.3. ԳՈՒՅՆ ԵՎ ՖՈՆ

CSS-ը հնարավորություն է տալիս հեշտությամբ սահմանել *WEB*-էջի տարրերի գույնն ու ֆոնը: Ընդ որում, ֆոնը կարելի է տալ նաև պատկերի տեսքով:

WEB-էջի տարրերի ու ֆոնի գույնը սահմանելու նպատակով *CSS*-ի մի շարք հատկանիշներ դիտարկենք:

color հատկանիշը նկարագրում է տարրի գույնը: Օրինակ, եթե անհրաժեշտ է փաստաթղթի *h1* մակարդակի բոլոր վերնագրերը սահմանել կարմիր, ապա *HTML*-ի <*h1*> տարրի համար կարելի է գրել.

```
h1 {color:#ff0000;}
```

background-color հատկանիշը նկարագրում է տարրի ֆոնի գույնը: Քանի որ *HTML* փաստաթղթի ողջ պարունակությունը ներառվում է <*body*> թէգում, ապա ամբողջ էջի ֆոնի գույնը փոփոխելու համար *background-color* հատկանիշը պետք է կիրառել <*body*> թէգի վրա: Այդ հատկանիշը կարելի է կիրառել նաև այլ տարրերի համար, այդ թվում վերնագրերի ու տեքստի վրա: Ստորև բերված օրինակում ֆոնի տարբեր գույներ են կիրառվել <*body*> և <*h1*> թէգերի վրա:

```
body {background-color:#FFCC66;}
h1 {color: #990000;background-color:#FC9804;}
```

Բերված օրինակում <*h1*>-ի համար երկու հատկանիշներ են կիրառվել (իրարից կետ-ստորակետով բաժանված):

background-image հատկանիշն օգտագործվում է ֆոնային պատկեր տեղադրելու համար: Դրա համար անհրաժեշտ է <*body*> թէգում կիրառել *background-image* հատկանիշը և նշել ֆոնային նկարի անունն ու հասցեն: Եթե պատկերը տեղադրելու եղանակ չտրվի, ապա պատկերը հորիզոնական և ուղղաձիգ ուղղություններով կրկնվելով՝ կծածկի ողջ էկրանը՝ սկսած վերին ձախ անկյունից:

Ստորև բերված օրինակում որպես ֆոնային պատկեր տեղադրվել է *nk1.gif* նկարը:

```
body {  background-color: #FFCC66;      background-image:url(nk1.gif);}
h1 {color: #990000;background-color: #FC9804;}
```

Ֆոնային պատկերը տեղադրելու գործընթացը կարգավորվում է *background-repeat* հատկանիշով: Աղյուսակ 3.1-ում այս հատկանիշի *4* արժեքներ են բերվել:

Օրինակ՝ ֆոնի պատկերը միայն հորիզոնական ուղղությամբ կրկնելու համար անհրաժեշտ է վերը բերված օրինակի կոդում *background-image* հատկանիշից հետո ավելացնել *background-repeat: repeat-x;* հրահանգը:

Աղյուսակ 3.1

Արժեքը	Նկարագրությունը
background-repeat: repeat-x	Պատկերը կրկնվում է հորիզոնական ուղղությամբ
background-repeat: repeat-y	Պատկերը կրկնվում է ուղղաձիգ ուղղությամբ
background-repeat: repeat	Պատկերը կրկնվում է հորիզոնական և ուղղաձիգ ուղղություններով
background-repeat: no-repeat	Պատկերը չի կրկնվում

background-attachment հատկանիշը հնարավորություն է տալիս ֆոնի պատկերն անշարժացնել *WEB*-էջում, կամ էջի պարունակությանը համընթաց շարժել: Աղյուսակ 3.2-ում այս հատկանիշի *2* արժեքներ են բերվել:

Աղյուսակ 3.2

Արժեքը	Նկարագրությունը
Background-attachment: scroll	Պատկերը տեղաշարժվում է էջի պարունակությանը համընթաց
Background-attachment: fixed	Պատկերն անշարժ է

background-position հատկանիշը հնարավորություն է տալիս սահմանելու ֆոնի պատկերի կոորդինատները (ֆոնի պատկերի դիրքը չսահմանելու դեպքում այն տեղադրվում է էկրանի վերին ձախ անկյունում): Հատկանիշի արժեքը կարելի է նշել չափման որևէ միավորով (*background-position: 3cm 4cm* օրինակի դեպքում պատկերը կտեղադրվի վերին ձախ անկյունից *3սմ* ձախ և *4սմ* ներքև), էկրանի լայնության տոկոսներով (*background-position: 40% 30%* օրինակի դեպքում պատկերը կտեղադրվի էկրանի վերին ձախ անկյունից էկրանի լայնության *40%*-ի չափով աջ և *30%*-ի չափով ներքև): Ֆոնի պատկերի կոորդինատները սահմանելիս կարելի է կիրառել հատկանիշի հետևյալ արժեքները. *top* (վերին), *bottom* (ստորին), *center* (կենտրոն), *left* (ձախ), *right* (աջ): Օրինակ՝ *background-position:right bottom* հրահանգի համաձայն պատկերը կտեղադրվի էկրանի ստորին աջ անկյունում:

background հատկանիշի միջոցով կարելի է մի շարք հատկություններ առավել համառոտ ձևակերպել: Օրինակ՝

```
background-color: #FFCC66;
background-image: url(nk1y.gif);
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

հրահանգները կարելի է միավորել հետևյալ հրահանգի մեջ.

```
background: #FFCC66 url(nk1.gif) no-repeat fixed right bottom;
```

Այս տարրն ունի հատկանիշների հետևյալ հերթականությունը.

- *[background-color]*
- *[background-image]*
- *[background-repeat]*
- *[background-attachment]*
- *[background-position]*

1. ***CSS*-ում տարրերի ու ֆոնի գույնը սահմանելու ինչ հատկանիշներ գիտեք:**
2. ***background-repeat* հատկանիշի ինչ արժեքներ գիտեք:**
3. ***CSS*-ը ֆոնի դիրքը սահմանելու ինչ եղանակներ գիտեք:**

Լաբորատոր աշխատանք 3.2
color և background հատկանիշների կիրառում

1. Ֆոնի համար որևէ նկար ընտրեք ու այն *fon.png* անվանումով պահպանեք ձեր ստեղծած *CSS* թղթապանակում:
2. *Start* գլխավոր մենյուի *Programs* ենթամենյուի *Accessories* ենթամենյուի *Notepad* հրամանով բացեք համանուն տեքստային խմբագրիչը:
3. Ընտրեք *Notepad* տեքստային խմբագրիչի մենյուի տողի *File* ենթամենյուի *Open* հրամանն ու բացեք ձեր ստեղծած *Style.css* ֆայլը:
4. Ավելացրեք հետևյալ կոդը.

```
body{
        background-image: url(fon.png);
        background-repeat:repeat;
}

table{
        width:900px;
        background-color:#66CDAA;
}
```

5. Խմբագրիչի մենյուի տողի *File* ենթամենյուի *Save* հրամանով պահպանեք խմբագրված փաստաթուղթը:
6. Մտեք որևէ բրաուզերի միջավայր: Ընտրեք *Notepad* խմբագրիչի միջավայրում ձեր ստեղծած *Index.html* ֆայլն ու մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարելով՝ էկրանին կտեսնեք ստորև բերված *WEB*-էջը՝ ձեր ընտրած ֆոնի նկարով.

7. Ավարտեք աշխատանքը՝ փակելով տեքստային խմբագրիչն ու բրաուզերը:

§3.4. ՏԵՔՍՏԻ ՁԵՎԱՎՈՐՈՒՄ

WEB-էջի տեքստը ձևավորելու գործընթացում տառաշարի պարամետրերը ճիշտ սահմանելը կարևոր նշանակություն ունի: *CSS*-ի այդ նպատակին ուղղված որոշ հատկանիշներ ուսումնասիրենք:

font-family հատկանիշը հնարավորություն է տալիս *WEB*-էջի կամ դրա առանձին տարրերի համար տառատեսակների ցանկալի առաջնահերթություն սահմանել: Եթե թվարկված տառատեսակներից առաջինը կայքից օգտվողի համակարգչում չկա, ապա փորձ է արվում կիրառել սահմանված առաջնահերթության ցուցակի հաջորդ հերթական տառատեսակը և այլն: Օրինակ.

h1 {font-family: arial, "times new roman", sans-serif;}

հրահանգի արդյունքում *<h1>* վերնագրերը կցուցադրվեն *arial* տառատեսակով: Կայքից օգտվողի համակարգչում այդ տառատեսակի բացակայու-

թյան դեպքում կօգտագործվի *times new roman* տառատեսակը: Առաջին երկու տառատեսակների բացակայության դեպքում՝ *sans-seif* ընտանիքի տառատեսակը: Տառատեսակի անվան մեջ բացատանիշերի առկայության դեպքում այն պետք է առնել չակերտների միջև:

font-style հատկանիշով սահմանվում է տառաշարի ձևը՝ *normal, italic* կամ *oblique*: Վերջինս կառուցվածքով *normal* ձևն է՝ որոշակի թեքությունով: Օրինակ՝

h2 {font-family: arial, serif; font-style:italic;}

հրահանգի արդյունքում *<h2>* վերնագրերը կցուցադրվեն *arial* կամ *serif* տառատեսակով և կընդունեն շեղ ձև:

font-weight հատկանիշով սահմանվում է տառաշարի հաստության աստիճանը: Հատկությունը կարող է ընդունել *normal* և *bold* արժեքներից որևէ մեկը: Օրինակ՝

p {font-family: arial, verdana, sans-serif;}

td {font-family: arial, verdana, sans-serif; font-weight:bold;}

Որոշ բրաուզերներ կարող են աշխատել նաև թվային արժեքների հետ: Այս դեպքում տառաշարի հաստության աստիճանը կարող է ընդունել *100, 200, 300, ..., 900* արժեքներից որևէ մեկը:

font-size հատկանիշով սահմանվում է տառաշարի չափը, որը տրվում է բացարձակ և հարաբերական մեծություններով:

Բացարձակ մեծությունը կարելի է նշել հետևյալ պահեստավորված բառերից որևէ մեկով.

- *xx-small* – փոքրագույն,
- *x-small* – շատ փոքր,
- *small* – փոքր,
- *medium* – միջին,
- *large* – մեծ,
- *x-large* – շատ մեծ,
- *xx-large* – մեծագույն:

Օրինակ՝ *p{font-size: x-large;}:*

Բացարձակ մեծությունը կարելի է նաև նշել՝ օգտագործելով *CSS*-ում չափի ընդունված միավորները՝ *px (փիքսել), pt (կետաչափ), in (դյույմ), cm (սանտիմետր), mm (միլիմետր)*: Օրինակ՝ *p{font-size: 14pt;}:*

Չափի հարաբերական մեծությունը կարելի է տալ հետևյալ պահեստավորված բառերից որևէ մեկով.

- *larger* – սահմանված չափից մեծ,
- *smaller* – սահմանված չափից փոքր,

կամ սահմանված չափի տոկոսով (%): Օրինակ՝ *p{font-size: 200%;}, h{font-size:larger;}:*

Այստեղ ևս *font* հատկանիշի միջոցով կարելի է մի շարք հատկանիշներ գրել կրճատ տեսքով: Օրինակ՝

```
p {
font-style: italic;
font-weight: bold;
font-size: 30px;
font-family: arial, sans-serif;
}
```

հրահանգները կարելի է միավորել մեկ հրահանգի մեջ.

```
p{font: italic bold 30px arial, sans-serif;}
```

Այժմ ծանոթանանք *CSS*-ի տեքստ ձևավորելու որոշ հնարավորություններին:

text-indent հատկանիշը հնարավորություն է տալիս սահմանել պարբերության առաջին տողի սկիզբը (նման հնարավորություն *HTML*-ում չկա):

Օրինակ՝ *p {text-indent: 2cm;}* հրահանգի արդյունքում բոլոր պարբերությունները կսկսվեն *2 սմ* խորքից:

text-align հատկանիշը հնարավորություն է տալիս սահմանել տեքստը հավասարեցնելու եղանակը. *left* (ըստ ձախ եզրի), *right* (ըստ աջ եզրի), *centred* (ըստ կենտրոնի) կամ *justify* (ըստ ձևաչափի):

Օրինակ՝ *td {text-align:left;}* հրահանգի արդյունքում աղյուսակի բջիջների պարունակությունները կհավասարեցվեն ըստ ձախ եզրի:

text-decoration հատկանիշը նախատեսված է տեքստը գեղարվեստորեն ձևավորելու համար: Այն կարող է ընդունել հետևյալ արժեքներից որևէ մեկը.

- *none* — գեղարվեստական ձևավորում չի իրականացվում,
- *underline* — յուրաքանչյուր տող ընդգծվում է ներքևից,
- *overline* — յուրաքանչյուր տող ընդգծվում է վերևից,
- *line-through* —յուրաքանչյուր տող պատկերվում է ջնջված,
- *blink* — տեքստը թարթում է:

Օրինակ՝

```
h1 {text-decoration:underline;}
h2 {text-decoration:overline;      }
```

հրահանգի արդյունքում *<h1>* մակարդակի վերնագրերը կլինեն ընդգծված ներքևից, իսկ *<h2>* վերնագրերը՝ վերևից:

letter-spacing հատկանիշը հնարավորություն է տալիս սահմանել տառերի միջև ցանկալի հեռավորությունները: Օրինակ՝ *p {letter-spacing: 4px;}* հրահանգի արդյունքում պարբերությունների տառերի միջև *4* փիքսել հեռավորություն կսահմանվի:

word-spacing հատկանիշը հնարավորություն է տալիս սահմանել տեքստի բառերի միջև ցանկալի հեռավորությունը: Օրինակ՝ *h1{word-spacing:2mm}* հրահանգի արդյունքում *<h1>* մակարդակի վերնագրերում բառերի միջև *2* մմ հեռավորություն կսահմանվի:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- *font-variant* հատկանիշը կարող է ընդունել *normal* կամ *small-caps* արժեքներից որևէ մեկը: Ընդ որում, *small-caps*-ի դեպքում ստեղների ստորին դիրքերում գտնվող տառերի փոխարեն փոքր չափի մեծատառեր կկիրառվեն:

1. ***CSS*-ում տառաշարի հետ աշխատելու ինչ հատկանիշներ գիտեք:**
2. ***CSS*-ում տեքստ ձևավորելու ինչ հատկանիշներ գիտեք:**

Լաբորատոր աշխատանք 3.3
Տեքստի ձևավորում

1. *Start* գլխավոր մենյուի *Programs* ենթամենյուի *Accessories* ենթամենյուի *Notepad* հրամանով բացեք համանուն տեքստային խմբագրիչը:
2. Ընտրեք *Notepad* տեքստային խմբագրիչի մենյուի տողի *File* ենթամենյուի *Open* հրամանը և բացեք ձեր ստեղծած *Style.css* ֆայլը:
3. Ֆայլի վերջում ավելացրեք հետևյալ կոդը.

```
h1{
        font-weight: bold;
        font-size: 30px;
        color: #ffffff;
        text-align:center;
   }
```

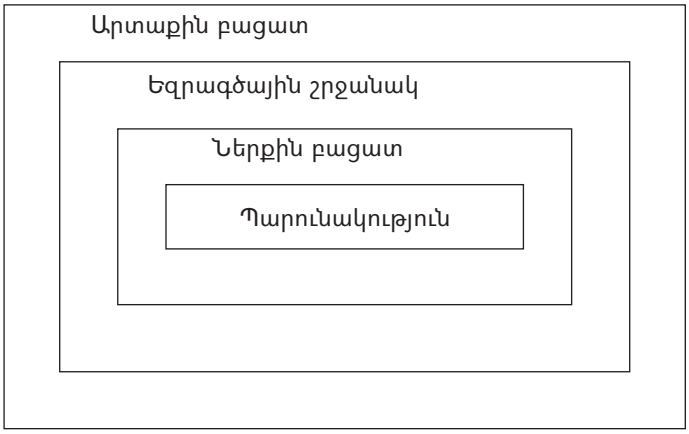
```
h2{
        font-style: italic;
        font-size: 22px;
        color: #006400;
    }
h3{
        font-size: 16px;
        color: green;
        text-align:center;
    }
p,li{
   font-size: 20px;
```

4. Խմբագրիչի մենյուի տողի *File* ենթամենյուի *Save* հրամանով պահպանեք խմբագրված փաստաթուղթը:
5. Մտեք որևէ բրաուզերի միջավայր: Ընտրեք *Notepad* խմբագրիչի միջավայրում ձեր ստեղծած *Index.html* ֆայլն ու մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարեք: Եթե ամեն ինչ ճիշտ եք կատարել, ապա էկրանին կտեսնեք ստորև բերված *WEB*-էջը.

6. Ավարտեք աշխատանքը՝ փակելով տեքստային խմբագրիչն ու բրաուզերը:

§3.5. ԲԼՈԿԱՅԻՆ ՄՈԴԵԼ

Բրաուզերները *HTML*-փաստաթղթի յուրաքանչյուր թէգ մշակում են որպես որոշակի բովանդակությամբ ուղղանկյուն տիրույթ, որը կարող է պարունակել տեքստ, պատկերներ, այլ թէգեր և այլն: *WEB*-էջը ձևավորվում է տարբեր հատկություններ ունեցող նման ուղղանկյուն բլոկներով (նկ. 3.2.): *WEB*-էջի ձևավորման այսպիսի մոդելն անվանում են բլոկային:



Նկ. 3.2. Տարրի բլոկը

Բլոկային մոդելում *արտաքին բացատը* դատարկ տարածք է, որը տարրերն իրարից տարանջատում է: Նման արտաքին տարածք է, օրինակ՝ պարբերություններն իրարից բաժանող միջակայքը:

Եզրագծային շրջանակը (այսուհետ՝ շրջանակ) տարրը շրջափակող եզրագիծն է:

Ներքին բացատը պարունակության և եզրագծի միջև առկա ազատ տարածքն է:

Պարունակությունը տարրի բովանդակությունն է:

Արտաքին և ներքին բացատները հատկություններով տարբերվում են: Եթե տարրի ֆոնը (*background*) սահմանվի, օրինակ՝ դեղին, ապա թե՛ բովանդակության տարածքը և թե՛ ներքին բացատը դեղին գույն կստանան, մինչդեռ արտաքին բացատը, դրսում գտնվելով, միշտ «թափանցիկ» կլինի (նկ. 3.3.):

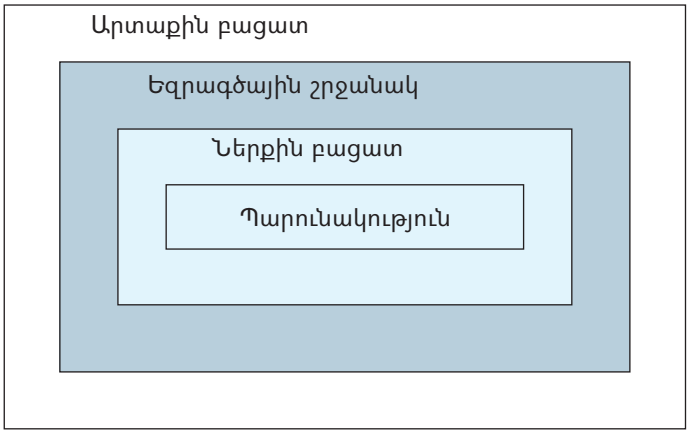
Արտաքին և ներքին բացատները դատարկ տարածություն են ստեղծում բլոկի «միջուկի»՝ տարրի շուրջը: Տարրի պարունակության և շրջանակի միջև բացատ ստեղծելու համար օգտագործվում է *padding*, իսկ արտաքին բացատի համար՝ *margin* հատկությունները: Յուրաքանչյուր տարրի համար առանձին-առանձին կարելի է նշել իր

արտաքին բացատի չափերը՝

margin-top	վերևից,
margin-right	աջից,
margin-bottom	ներքևից,
margin-left	ձախից

և *ներքին բացատի չափերը՝*

padding-top	վերևից,
padding -right	աջից,
padding -bottom	ներքևից,
padding -left	ձախից:



Նկ. 3.3. Տարրի բլոկի օրինակ

Օրինակ՝ արտաքին բացատի չափերը կարելի է նշել հետևյալ կերպ.

```
margin-top: 20px;
margin-right: 15px;
margin-bottom: 20px;
margin-left: 30px;
```

Չափերի նույն տվյալները կարելի է սահմանել նաև հետևյալ կրճատ գրելաձևով.

```
margin: 20px 15px 20px 30px;
```

Ներքին բացատի չափերը *padding*-ի միջոցով նույնպես սահմանվում է

```
padding-top: 20px;
padding-right: 30px;
padding-bottom: 40px;
padding-left: 50px;
```

օրինակի համաձայն. այստեղ ևս կարելի է կրճատ գրելաձև կիրառել.

```
padding: 20px 30px 40px 50px;
```

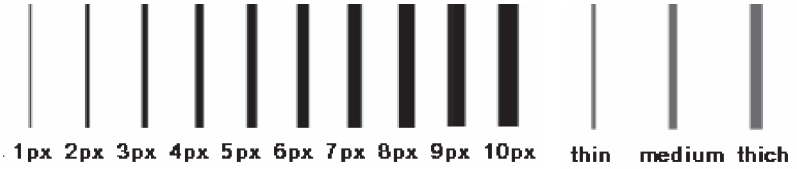
Օրինակ՝ *h1* և *h2* մակարդակի վերնագրերի համար սահմանենք ֆոնի գույներն ու վերնագրերի շուրջը եղած դաշտերի (ներքին բացատի) չափերը.

```
h1 {
background: yellow;
padding: 20px 30px 40px 50px;
}
h2 {
background: orange;
padding-left:120px;
}
```

Տարրի բլոկային մոդելում շրջանակը տարվում է սովորական գծով և տարրը բոլոր կողմերից եզրագծելով՝ նշում է դրա սահմանները:

CSS-ում շրջանակները տրվում են *border* հատկանիշի միջոցով, որտեղ նշվում են տարվող գծի *հաստությունը*, *ձևն* ու *գույնը*՝ համապատասխանաբար *border-width*, *border-style* և *border-color* հատկանիշներով:

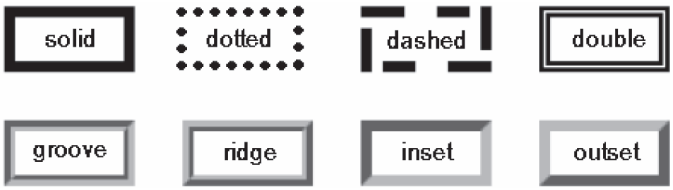
border-width հատկանիշը կարող է ընդունել *thin*, *medium*, *thick* արժեքներից որևէ մեկը կամ որևէ թվային արժեք՝ փիքսելներով (նկ. 3.4.):



Նկ. 3.4. Շրջանակի գծի հաստության օրինակներ

Շրջանակի համար տրվող *border-color* հատկանիշը կարող է գույնի համար ընդունելի որևէ արժեք ստանալ, օրինակ, *“#123456”*, *“rgb(123,123,123)”* կամ *“yellow”*:

Շրջանակի *border-style* հատկանիշով սահմանվող գծի տիպը կարող է ընդունել *solid*, *dotted*, *dashed*, *double*, *groove*, *ridge*, *inset* կամ *outset* արժեքներից որևէ մեկը (նկ. 3.5.): Եթե շրջանակի տիպը չի սահմանվել, ապա *border-style* հատկանիշը կընդունի *solid* արժեքը:



Նկ. 3.5. Շրջանակի տիպեր

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Եթե *margin* և *padding* հատկանիշներում նշվի իրարից բացատանիշով բաժանված երկու արժեք, ապա դրանցից առաջինը կվերագրվի դաշտի վերին և ստորին, իսկ երկրորդը՝ ձախ և աջ չափերին: Իսկ եթե նշվի երեք արժեք, ապա դրանցից առաջինը կվերագրվի դաշտի վերին, երկրորդը՝ ձախ և աջ, երրորդը՝ ստորին չափերին:

1. **Ի՞նչ է բլոկային մոդելը:**
2. ***CSS*-ում *margin* հատկանիշով ի՞նչ պարամետրեր են սահմանվում:**
3. ***CSS*-ում *padding* հատկանիշով ի՞նչ պարամետրեր են սահմանվում:**
4. **Շրջանակի ի՞նչ պարամետրեր գիտեք:**

§3.6. ՏԱՐՐԻ ՉԱՓԻ ՈՒ ԴԻՐՔԻ ՍԱՀՄԱՆՈՒՄԸ

HTML-ում սովորեցինք աղյուսակների օգնությամբ հնարավորինս կարգավորել *WEB*-էջի դիզայնը: Այդ առումով *CSS*-ը առավել մեծ հնարավորություններ է ընձեռում: Այստեղ հնարավորություն ունենք ճշգրտորեն սահմանելու տարրերի չափերն ու դիրքը:

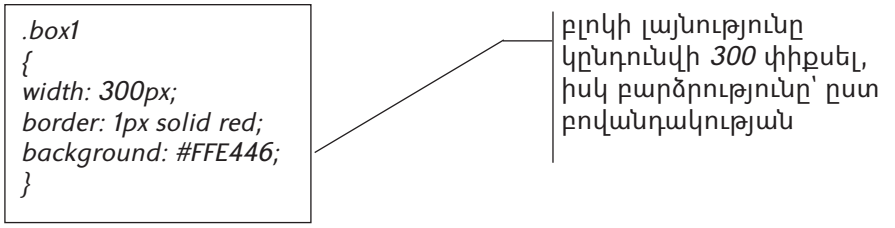
Տարրի *լայնությունը* կարելի է սահմանել *width*, իսկ *բարձրությունը՝ height* հատկանիշով: Այս հատկանիշները կարող են ընդունել *auto* արժեք, որի դեպքում տարրի չափը որոշվում է իր բովանդակությամբ (տարրի չափը սահմանելու այս ձևն է ընտրվում, եթե չափը բացահայտ չի տրվում):

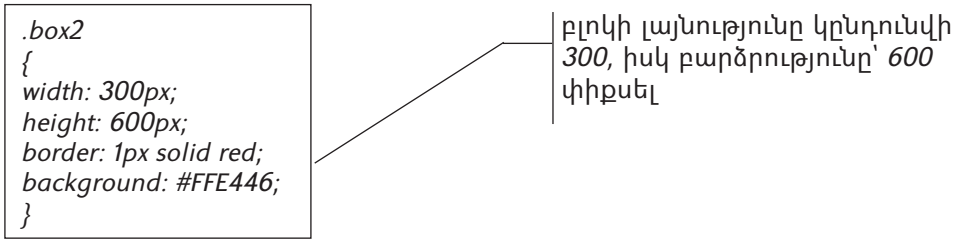
width և *height* հատկանիշների համար որպես չափման միավոր կարելի է կիրառել.

% – տարրի չափը տրվում է նախասահմանված չափի տոկոսներով,

px – տարրի չափը տրվում է փիքսելներով (կարելի է նշել նաև *CSS*-ում ընդունված չափման այլ միավորներ):

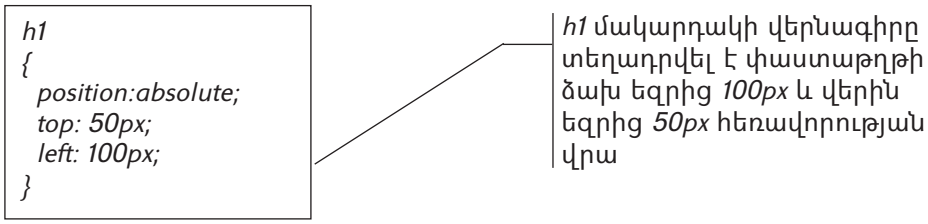
Դիտարկենք հետևյալ օրինակները.





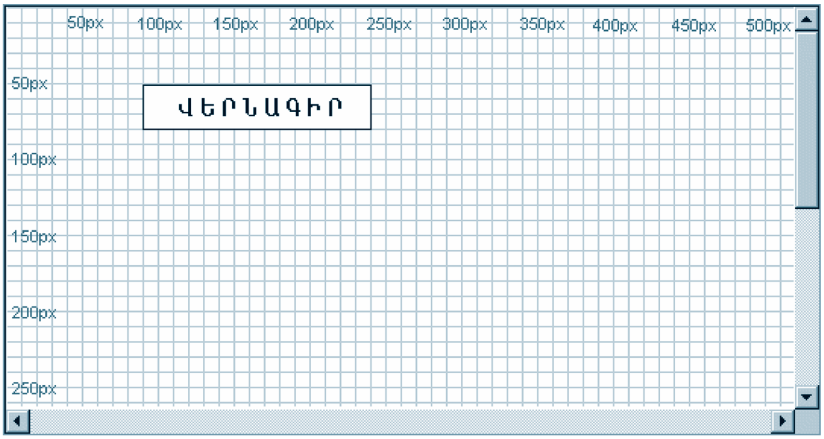
WEB-էջում տարրի տեղակայման ճշգրիտ դիրքը սահմանելու համար *CSS*-ը լայն հնարավորություններ է ընձեռում: Դիրքի սահմանման ռեժիմը կառավարում է *position* հատկանիշը, որի օգնությամբ ընտրվում է տարրի դիրքի հաշվարկման եղանակը:

Դիրքի *բացարձակ սահմանման* դեպքում *position* հատկանիշը պետք է *absolute* արժեքն ունենա: Այնուհետև *left, right, top* և *bottom* պահեստավորված բառերի օգնությամբ նշվում են տարրի կոորդինատները՝ հաշվարկված բրաուզերի պատուհանի եզրերից: Դիրքի բացարձակ սահմանման դեպքում տարրը հեռացվում է իր հիմնական դիրքից և տեղադրվում ըստ նոր կոորդինատների: Օրինակ՝



CSS-ի ստորև բերված կոդի արդյունքում կստանանք նկ. 3.6.-ում պատկերվածը (բրաուզերի պատուհանն ընդունվել է որպես կոորդինատների համակարգ):

Դիրքի *հարաբերական սահմանման* դեպքում *position* հատկանիշը պետք է *relative* արժեքն ունենա: Ի տարբերություն դիրքի բացարձակ սահմանման՝ տարրի նոր կոորդինատները հաշվարկվում են՝ որպես կոորդինատների սկզբնակետ ընդունելով նախկին կոորդինատները:



Նկ. 3.6. Տարրի տեղադրումը բրաուզերի պատուհանին, որը բերվել է որպես կոորդինատների համակարգ

```
#dog
{
  position:relative;
  left: 300px;
  bottom: 100px;
}
```

պատկերը տեղաշարժվել է իր նախկին դիրքից *300* փիքսել ձախ և *100* փիքսել ներքև

```
#dog
{
 position:relative;
  top: 200px;
  left: 100px;
}
```

պատկերը տեղաշարժվել է նախկին դիրքից *200* փիքսել վերև և *100* փիքսել ձախ

Եթե անհրաժեշտ է փաստաթղթում տարրի դիրքն անշարժացնել այնպես, որ անցավազքի ընթացքում չտեղաշարժվի, ապա *position* հատկանիշը պետք է *fixed* արժեքն ունենա: Օրինակ՝

```
h1
{
  position:fixed;
  top: 200px;
  left: 100px;
}
```

պատկերը սևեռվել է փաստաթղթի վերին սահմանից *200px* և ձախ սահմանից *100px* հեռավորության վրա

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

- Եթե տարրը գերազանցում է նշված չափը, ապա որոշ բրաուզերներում *width* և *height* հատկանիշները կարող են ընդունել *auto* արժեքը:

1. ***CSS*-ում տարրի չափը սահմանելու ինչ ձևեր գիտեք:**
2. ***CSS*-ում տարրի դիրքը սահմանելու ինչ եղանակներ գիտեք:**
3. ***CSS*-ի օգնությամբ ինչպե՞ս կարելի է տարրի դիրքը սևեռել *WEB*-էջում:**

Լաբորատոր աշխատանք 3.4
Տարրի դիրքի և չափի սահմանում

1. *Start* գլխավոր մենյուի *Programs* ենթամենյուի *Accessories* ենթամենյուի *Notepad* հրամանով բացեք համանուն տեքստային խմբագրիչը:
2. Ընտրեք *Notepad* տեքստային խմբագրիչի մենյուի տողի *File* ենթամենյուի *Open* հրամանն ու բացեք ձեր ստեղծած *style.css* ֆայլը:
3. Ֆայլում ավելացրեք գույնով ընդգծված հրահանգները:

```
table, td{border: 1px solid green;
}
body{
        background-image: url(fon.png);
        background-repeat:repeat;
}
table{
        width:900px;
        background-color:#66CDAA;
        margin:auto;
}
td{
   vertical-align:top;
}
```

```
h1{
        font-weight: bold;
        font-size: 30px;
        color: #ffffff;
        text-align:center;
}
h2{
        font-style: italic;
        font-size: 22px;
        color: #006400;
}
h3{
        font-size: 16px;
        color: green;
        text-align:center;
}
p,li{
   font-size: 20px;
}
img{height:120px;
   width:100px;
        margin:20px;
        float:left;
}
```

4. Խմբագրիչի մենյուի տողի *File* ենթամենյուի *Save* հրամանով պահպանեք խմբագրված փաստաթուղթը:
5. Մտեք որևէ բրաուզերի միջավայր: Ընտրեք *Notepad* խմբագրիչի միջավայրում ձեր ստեղծած *Index.html* ֆայլն ու մկնիկի ձախ սեղմակի կրկնակի սեղմում կատարելով՝ էկրանին կտեսնեք ստորև բերված *WEB*-էջը.

6. Ավարտեք աշխատանքը՝ փակելով տեքստային խմբագրիչն ու բրաուզերը:

§3.7. ԽՈՐՀՈՒՐԴՆԵՐ *WEB*-ԿԱՅՔԻ ԿԱՌՈՒՑՎԱԾՔԻ ԵՎ ՏԵՂԱԿԱՑՄԱՆ ՎԵՐԱԲԵՐՅԱԼ

Կայքի ստեղծման փուլում հնարավոր խնդիրներից խուսափելու համար խորհուրդ է տրվում.

- *WEB*-կայքը ստեղծելուց առաջ մտածեք կայքի այցելուների մասին: Սրանից կախված պետք է որոշել էջերի արտաքին տեսքն ու ինֆորմացիայի տրման ոճը: Եթե կայքը ձեր մերձավորների համար է ստեղծվում, այն կարող է լինել պարզ և ուրախ, ինչպես նաև կարող է անձնական բնույթի ինֆորմացիա ներառել: Եթե այն գործնական բնույթ կրող կայք է, ապա կայքում անհրաժեշտ է տեղադրել այնպիսի հղումներ, որոնք կհետաքրքրեն այցելուներին:
- Այլ անձից փոխառված ինֆորմացիան ձեր կայքում տեղադրելու համար նախ ստացեք նրա թույլտվությունն ու այդ ինֆորմացիայից օգտվելիս հղում արեք բնօրինակին:
- Կայքն ավելի հարուստ կլինի, եթե թեմային առնչվող արտաքին հղումներ ունենա:
- Կայքը պետք է այնպես կառուցել, որ այն էջերի միջև հնարավորինս պարզ կապեր ունենա: Պետք է հստակ նշել, թե որ հղումներն են նախատեսված կայքի տարածքում տեղաշարժվելու համար և որոնք են Համացանցի այլ կայքեր տանում:
- Գրաֆիկական և մուլտիմեդիա ինֆորմացիաները *WEB*-էջը կարող են առավել գրավիչ դարձնել, սակայն դրանց քանակը պետք է չափավոր լինի, քանի որ էջի ծավալի մեծացմանը զուգընթաց դանդաղեցվում է էջի բեռնավորման գործընթացը: Հիշեք, որ դեռևս Համացանցից օգտվողների մի մասը Համացանց մտնելու համար ոչ արագագործ մոդեմներից է օգտվում:
- *WEB*-էջերը Համացանցում տեղադրելուց առաջ պետք է անպայման թեստավորվի. դիտվի համակարգչի վրա, ստուգվի հղումների աշխատանքը, տեքստի և գրաֆիկայի տեղաբաշխումը, դիտվի տարբեր բրաուզերների վրա: Խնդրեք ձեր ընկերներին՝ օգնել այդ գործում:
- Համացանցում աշխատելիս մի խախտեք էթիկայի ընդունված կանոնները, այլապես կարող եք ոչ միայն կորցնել ձեր կայքն ունենալու իրավունքը, այլև խնդիրներ ունենալ իրավապահների հետ:
- Կայքի գլխավոր էջում տեղադրեք ձեր էլեկտրոնային հասցեն և այցելուներին խնդրեք ձեզ ուղարկել իրենց կարծիքները. այսպիսով օգտակար խորհուրդներ ստանալու հնարավորություն կունենաք:
- Անընդհատ թարմացրեք ձեր կայքը՝ այցելուներ չկորցնելու համար:

Ձեր ստեղծած *WEB*-կայքը խորհուրդ ենք տալիս պահպանել ստորև բերված տարածքներից որևէ մեկում:

- *www.sites.google.com* – *Google*-ը հնարավորություն է տալիս պարզ կայքեր ստեղծել և տեղ է տրամադրում ծավալուն կայք տեղակայելու համար:
- *www.geocites.com* – *Yahoo*-ն հնարավորություն է տալիս *Yahoo*-ի *GeoCites* կայքի օգնությամբ *WEB*-էջեր ստեղծել:
- *www.apple.com/mobileme* – *MobileMe*-ն *Macintosh* համակարգիչ կամ *iPhone* հեռախոս ունեցողներին հնարավորություն է տալիս ինչպես *WEB*-էջեր ստեղծել, այնպես էլ օգտվել մի շարք այլ ծառայություններից:
- *www.ning.com* – *Ning*-ը նախատեսված է ոչ միայն *WEB*-կայքի, այլև սեփական սոցիալական կայք ստեղծելու համար:
- *www.narod.yandex.ru* – *Narod*-ը անվճար հարմար գործիքաշար է առաջարկում, ինչը հնարավորություն է տալիս սեփական *WEB*-կայքում լուսանկարչական ալբոմներ, անհատական ֆորում և այլ ինտերակտիվ տարրեր տեղադրել:

ՕԳՏԱԿԱՐ Է ԻՄԱՆԱԼ

➢ *WEB*-կայքի տեղադրման մասին տեղեկատվություն կարելի է ստանալ *www.hostikus.ru, www.terrafiles.ru, www.hostland.su* կայքերում:

1. ***WEB*-կայքի կառուցվածքի մասին մի քանի խորհուրդներ թվարկեք:**
2. ***WEB*-կայքի տեղակայման մի քանի տարածքներ թվարկեք:**

Բովանդակություն

Ս. Ս. ԱՎԵՏԻՍՅԱՆ, Ս. Վ. ԴԱՆԻԵԼՅԱՆ

ԻՆՖՈՐՄԱՏԻԿԱ

12-րդ դասարան

ՀԱՆՐԱԿՐԹԱԿԱՆ ԱՎԱԳ ԴՊՐՈՑԻ

ԸՆԴՀԱՆՈՒՐ ԵՎ ՀՈՒՄԱՆԻՏԱՐ ՀՈՍՔԵՐԻ ՀԱՄԱՐ

Խմբագիր՝ Արտակ Սուրենի Ոսկանյան

Սրբագրիչ՝ Անահիտ Պապյան
Ձևավորումը՝ Նվարդ Հայրապետյանի
Շապիկի ձևավորումը՝ Աննա Կարապետյանի
Շարվածքը՝ Ալվարդ Ավետիսյանի

Պատվեր՝ 1148: Տպաքանակ՝ 21135:
Թուղթը՝ օֆսեթ: Չափսը՝ 70x100/16: 8 տպ. մամուլ:
Տառատեսակը՝ GHEA Arpi Sans:

Տպագրված է **«Տիգրան Մեծ»** հրատարակչություն ՓԲԸ տպարանում